Degraded Text Recognition of Gurmukhi Script

A Thesis

Submitted in fulfilment of the requirements for the award of the degree of

Doctor of Philosophy

Submitted by

Manish Kumar

(Registration No. 9000351)

Under the supervision of

Dr. R. K. Sharma
Professor, Thapar University
Patiala.

Dr. G. S. Lehal
Professor, Punjabi University
Patiala.

Department of Computer Science and Engineering
Thapar University
Patiala–147004 (Punjab) India.

March 2008

CERTIFICATE

Certified that the thesis entitled **DEGRADED TEXT RECOGNITION OF GURMUKHI SCRIPT**, which is being submitted by **Mr. Manish Kumar** (Registration No. 9000351), to the Department of Computer Science and Engineering (DCSE), Thapar University, Patiala, in fulfilment of the requirements for the award of the degree of **DOCTOR OF PHILOSOPHY,** is a record of bonafide research work carried out by him under our guidance and supervision. The matter presented in this thesis has not been submitted either in part or full to any other University or Institute for award of any degree.

(R. K. Sharma)
Professor and Dean,
School of Mathematics and Computer Applications
Thapar University, Patiala–147004 (INDIA).
Supervisor

(G. S. Lehal)
Professor and Head,
Department of Computer Science
Punjabi University, Patiala–147004 (INDIA).
Supervisor

Dated: March 18, 2008

Acknowledgements

I would like to express in these lines my deep gratitude to those people who have helped me in the accomplishment of this Ph. D. thesis.

First and foremost, I would like to thank my supervisors, Dr. R. K. Sharma and Dr. G. S. Lehal for their expertise, guidance, enthusiasm and patience. These were invaluable contributing factors to the successful completion of this work.

I am deeply grateful to Dr. G. S. Lehal for introducing me with Dr. R. K. Sharma and making this research possible.

I would also like to thank Dr. B. B. Chaudhuri, ISI Kolkata, Dr. U. Pal, ISI Kolkata and Dr. Chandan Singh, Punjabi University Patiala for their valuable suggestions.

I would also like to give special thanks to my parents and my wife and son for their inspiration and encouragement. Without their support this research would not have been possible.

Finally, I would like to thank all academic, administrative and technical staff from the Thapar and Punjabi University for their encouragement, assistance and friendship throughout my candidature.

Manish Kumar

Manish Kumar

Abstract

Character recognition is one of the important subjects in the field of Document Analysis and Recognition (DAR). Character recognition can be performed on printed text or handwritten text. Printed text can be from good quality documents or degraded documents. There are several kinds of degradations in almost every script of the world. The list of normally found degradations in any printed script includes touching characters, broken characters, heavy printed characters (self touching), faxed documents, typewritten documents and backside text visible documents. The problem of touching characters commonly exists in all the degraded documents containing these kinds of degradations. Hence, it is the need of the time to cope with the problem of touching characters to make an Optical Character Recognition (OCR) for degraded text.

Researchers involved in recognition of good quality printed text in different scripts around the world have reported drastic decrease in recognition accuracy due to presence of touching characters in the text. Research and experiments have shown that performance breakdown of commercial document recognition system under real application situations is caused mainly due to the difficulty in dealing with touching characters that are abundant in documents as a result of document degradations. Touching characters make it difficult to correctly segment character images for individual classification, and therefore, pose severe difficulty to conventional document recognition systems that are critically dependent on character segmentation. The problem of heavy printed characters also decreases recognition accuracy. A document containing touching characters generally contains heavily printed characters also.

Objective of this work is to seek new approaches to degraded document recognition of printed Gurmukhi script containing touching characters and heavily printed characters. OCR algorithms can achieve good recognition rates (near 99%) on images with little degradation. However, recognition rates drop to 70% or even lower when image degradations are present. Typical pages of text have more than 2000 characters per page. Therefore, an error rate of 30% results in more than 600 mistakes. Before the mistakes can be corrected, they must be located, making the correction process even more tedious.

Currently, there is no software available for OCR of degraded printed Gurmukhi script, in particular, and other degraded printed Indian language scripts, in general. There is a dire need for the OCRs of Indian language scripts as people working in Indian language scripts are denied the opportunity of converting scanned images of degraded machine-printed or handwritten text into a computer processable format. This work is the first attempt towards the development of an OCR for recognising degraded documents of printed Gurmukhi script. This work can lead towards the development of OCRs for other Indian language scripts such as Devanagari, Bangla *etc.* that are structurally similar to Gurmukhi script.

This thesis is divided into seven chapters. First chapter introduces the process of OCR and various phases of OCR like pre-processing, segmentation, feature extraction, classification and post-processing. Problems in text recognition due to presence of degraded text in a script, in general, and in Gurmukhi script, in particular, have been discussed. The need of an OCR for recognising degraded printed documents containing touching and heavily printed characters of Indic scripts has also been discussed.

In second chapter, a comprehensive and exhaustive review of the literature for various methods used for segmenting machine-printed scripts and degraded printed scripts have been discussed. Also, various methods used in literature for feature extraction and classification have been discussed. A detailed survey on Indian script recognition systems has also been carried out. We have also discussed the work done by various researchers for recognising degraded text.

Third chapter starts with study of importance of degradation models used for recognising degraded data. The properties of Gurmukhi script and other Indian scripts have been discussed. Various kinds of degradations in degraded printed Gurmukhi script are also presented in this chapter. The problems associated with recognition of printed Gurmukhi script documents containing touching characters, broken characters, heavy printed characters, faxed data, typewritten data, and backside text visible characters have also been discussed. The reason of occurrence, comparison of each kind of degradation with corresponding degradations in Roman script and some possible solutions have been discussed for each kind of degradations.

Chapter 4 consists of algorithms proposed for segmenting touching characters in degraded printed Gurmukhi script. In the first algorithm, a method has been proposed to

segment horizontally overlapping lines and associating broken components of a line (small strips) with their respective lines. Various types of strips have been identified in good quality as well as degraded printed documents of Gurmukhi script with percentage of occurrence of each type of strip in document database. The modified version of the algorithm has been proposed for segmenting horizontally overlapping lines of multiple sized texts. The proposed algorithm had also been successfully tested on other Indian scripts like Devanagari, Bangla, Gujarati, Kannada, Tamil, Telugu and Malayalam for segmenting the horizontally overlapping lines and associating broken components of a line with their respective lines. Segmentation accuracy of 95% to 99.7% has been achieved with the use of these algorithms for various scripts.

Further, different categories of touching characters in all the three zones (upper, middle and lower zone) of degraded printed Gurmukhi script has been identified on the basis of structural properties of Gurmukhi script. In another algorithm, a method for segmenting touching characters in upper zone has been proposed with an accuracy of 92%. The algorithm is based upon the structural properties like concavity and convexity of sub-symbols (connected components of a character) in upper zone. This algorithm successfully segments highly touching characters and also segments small characters such as *bindī* from other characters. One more algorithm has been developed for segmenting touching characters in middle zone. This algorithm is very effective for segmenting touching sub-symbols with 91% accuracy. The solution has also been proposed for segmenting touching sub-symbols in lower zone. These are new algorithms that have been proposed by us for segmenting degraded text in Gurmukhi script. It is also shown that such algorithms can also be adopted for segmentation of degraded text in Devanagari, Bangla and other Indic scripts.

In Chapter five, structural and statistical features used for extracting the features of segmented characters of degraded printed Gurmukhi script have been discussed. Structural features like presence of sidebar, presence of half sidebar, presence of headline, number of junctions with headline, number of junctions with baseline, aspect ratio, left and right profile direction codes, top and bottom profile direction codes and transition features have been used. Another useful structural feature, named, Directional Distance Distribution (DDD) has been used, which is based upon the distance of nearest black/white pixel in eight directions for each white/black pixel in the input binary array. We have given the detection accuracy of

each structural feature also. Additional assumptions have been proposed to improve the detection accuracy. Some of the statistical features including zoning, Zernike moments, Orthogonal Fourier Mellin (OFM) moments have been used for extracting the features. A detailed performance analysis on various options of each structural and statistical feature for all the three zones has been carried out.

In Chapter six, various classifiers used for recognition of text have been discussed. We have developed a corpus for degraded printed Gurmukhi script OCR. A number of documents from various sources like newspapers, old and new books, magazines printed on low quality paper, computer printouts, faxed documents and typewritten documents were collected and scanned, which were used for training and testing purpose. Most commonly used classifiers such as *k*-Nearest Neighbor (*k*-NN), Support Vector Machines (SVM), and Artificial Neural Networks (ANN) have been used for recognition purpose. We have used MATLAB 7.2 for implementing *k*-NN and SVM classifiers. NeuNet Pro 2.3 has been used for implementing ANN. We have obtained an accuracy of 92.54% in recognition of degraded printed Gurmukhi script characters using SVM classifier.

Finally, chapter seven presents the inferences drawn from the results of the various experiments conducted in this thesis. Also, some pointers to the future research on the topic under consideration in this thesis are discussed briefly in this chapter.

Contents

CERTIFICATE…….. i

ACKNOWLEDGEMENTS……... ii

ABSTRACT……..... iii

LIST OF FIGURES……... xi

LIST OF TABLES……..... xvii

1. INTRODUCTION…….. 1

1.1 Historical Background……... 2

1.2 Character Recognition……... 3

1.3 Components of an OCR……. 4

1.3.1 Digitization……... 4

1.3.2 Pre-processing……... 5

1.3.3 Segmentation……..... 6

1.3.4 Feature extraction……. 6

1.3.5 Classification……..... 6

1.3.6 Post-processing …… 7

1.4 Need for Study……..... 7

1.5 Motivation……..... 8

1.6 Objectives of this Research……..... 11

1.7 Major Contributions and Achievements……... 12

1.8 Assumptions……..…… 13

1.9 Organization of Thesis…….. 13

2. REVIEW OF LITERATURE……... 14

2.1 Segmentation……...……... 15

2.1.1 Segmentation of touching characters…… 18

2.1.2 Segmentation of touching numerals…….. 21

2.1.3 Segmentation of touching characters in Indian scripts……..... 22

2.2 Feature Extraction……..... 24

2.2.1 Statistical or global features……..... 25

2.2.2 Structural features……..... 27
2.2.3 Series expansion coefficients…… 28
2.3 Classification…….. 28
2.3.1 Template matching……..... 29
2.3.2 Syntactic or structural methods……..... 29
2.3.3 Statistical methods…… 30
2.3.4 Artificial neural networks……..... 30
2.3.5 Kernel methods……..... 31
2.3.6 Hybrid classifiers……..... 31
2.4 Indian Script Recognition……..... 32
2.4.1 Devanagari……..... 32
2.4.2 Bangla……..... 34
2.4.3 Tamil……..... 36
2.4.4 Telugu……... 36
2.4.5 Oriya……. 37
2.4.6 Gurmukhi……..... 37
2.4.7 Gujarati…….. 40
2.4.8 Kannada…… 40
2.5 Degraded Text Recognition……... 41
2.6 Degraded Text Recognition: A Challenge in OCR Research…… 43

3. DIFFERENT KINDS OF DEGRADATIONS……..... 45
3.1 Characteristics of Gurmukhi Script…… 47
3.2 Characteristics of other Indian Scripts……..... 49
3.3 Touching Characters……..... 50
3.4 Broken Characters……..... 53
3.5 Heavy Printed Characters……... 56
3.6 Faxed Documents……... 57
3.7 Typewritten Documents……..... 59
3.8 Documents containing Backside Text Visible……... 60
3.9 Discussion…….. 61

4. SEGMENTATION …… 62
4.1 System Overview……..... 62
4.2 Pre-processing……..... 64
4.3 Segmentation Process……..... 64
4.4 Line Segmentation……..... 66
4.4.1 Segmentation of overlapping line of uniform size…… 72
4.4.2 Segmentation of overlapping line of uniform size in other Indian scripts…… 77
4.5 Segmentation of Overlapping Lines of Different Sized Text…… 92
4.6 Word and Zone Segmentation……..... 95
4.7 Character Segmentation……. 96
4.7.1 Categories of touching sub-symbols in upper zone…….. 98
4.7.2 Categories of touching sub-symbols in middle zone…… 101
4.7.3 Categories of touching sub-symbols in lower zone…….. 104
4.8 Segmentation of Touching Sub-symbols in Upper Zone……... 106
4.9 Segmentation of Touching Sub-symbols in Middle Zone……..... 112
4.9.1 Solution for segmenting touching sub-symbols falling in first category……..... 115
4.9.2 Solution for segmenting touching sub-symbols falling in second category……..... 117
4.9.3 Solution for segmenting touching sub-symbols falling in third category……... 117
4.9.4 Solution for segmenting touching sub-symbols falling in fourth category…… 118
4.10 Segmentation of Touching Sub-symbols in Lower Zone…… 119
4.11 Results and Discussion……. 120

5. FEATURE EXTRACTION ……..... 125
5.1 Structural Features……..... 126
5.1.1 Additional assumptions……..... 134

5.2 Statistical Features……..... 135
5.2.1 Zoning……..... 135
5.2.2 Moments……..... 136
5.2.2.1 Zernike moments……..... 137
5.2.2.2 Orthogonal Fourier Mellin Moments……..... 138
5.3 Performance Analysis of Structural and Statistical features……..... 140
5.3.1 Standard feature values of middle zone sub-symbols..... 140
5.3.2 Standard feature values of upper and lower zone sub-symbols..... 144
5.3.3 Performance analysis of features for middle zone sub-symbols..... 146
5.3.4 Performance analysis of features for upper and lower zone characters..... 152
5.4 Size of Feature Set……..... 153
5.5 Discussion…….. 154

6. CLASSIFICATION……..... 156
6.1 Classifiers……..... 156
6.1.1 Neural networks ……..... 157
6.1.2 Nearest neighbor classifier…… 160
6.1.3 SVM……..... 160
6.2 Merging Sub-symbols……..... 162
6.3 Implementation details……..... 163
6.4 Experiments…….……..... 165
6.4.1 Sample images……..... 165
6.5 Results and Discussion……..……. 171

7. CONCLUSIONS……..……..... 186
7.1 Contributions of the Work……..... 187
7.2 Future Work…….……..... 188
BIBLIOGRAPHY…….. 189
PUBLICATIONS BASED ON THE WORK PRESENTED IN THIS THESIS…. 204

List of Figures

1.1	Hierarchy of character recognition problems……..	4
1.2	Block diagram for a typical OCR application…….....	5
1.3	Examples of characters with different degradations that lead to common OCR errors: (a) true characters, (b) characters with negative edge displacement, (c) characters with positive edge displacement.……...	9
1.4	Degraded printed Gurmukhi script document containing touching characters and the heavily printed characters.…….....	10
1.5	Degraded printed Gurmukhi script document containing touching characters and the heavily printed characters ……....	10
2.1	Touching characters in Roman script (Dunn *et al.* [40])…….....	19
2.2	Water reservoirs (shown by dotted lines) in single and touching Oriya characters……	24
2.3	The Interactive model of text recognition system proposed by Hong [51]…….....	42
3.1	Characters and symbols of Gurmukhi script……...	47
3.2	Gurmukhi script word……	48
3.3	Gujarati script word…….....	49
3.4	Touching characters in Gurmukhi script…….....	50
3.5	Touching characters in three zones (touching parts encircled): (a) upper zone characters touching each other, (b) upper zone characters touching with middle zone characters, (c) middle zone characters touching with each other, (d) middle zone characters touching with lower zone characters, (e) lower zone characters touching with each other……...	52
3.6	Document containing touching characters in neighboring lines.....	52
3.7a	Broken characters in Gurmukhi script…….....	53
3.7b	Extremely broken characters in Gurmukhi script……...	54
3.8	Heavily printed characters in Gurmukhi script……...	56
3.9	Words taken from faxed documents of Gurmukhi script……	58

3.10	Words taken from typewritten documents of Gurmukhi script….	59
3.11	Gurmukhi script document containing backside text visible……..	60
4.1	System model for recognizing degraded printed Gurmukhi text….....	63
4.2	Segmentation process: (a) input document, (b) text regions extracted from input document, (c) segmented lines, (d) segmented words, (e) segmented characters……	66
4.3	Horizontal projection of Gurmukhi script document resulting over segmentation……...	67
4.4	Horizontal projection of Gurmukhi script document resulting under segmentation…….....	68
4.5	Various types of strips in degraded printed Gurmukhi script…….	70
4.6	Line boundaries identified by applying Algorithm 4.1 on Figure 4.5…….....	75
4.7	Strips containing horizontally overlapping lines…….....	76
4.8	Line boundaries identified after applying Algorithm 4.1 on Figure 4.8.....	76
4.9a	Strips in printed Devanagari script document…….....	83
4.9b	Line boundaries identified using proposed Algorithm 4.2…….	84
4.10a	Strips in printed Bangla script document…….....	84
4.10b	Line boundaries identified using proposed Algorithm 4.2…….	85
4.11a	Strips in printed Gujarati script document……..	85
4.11b	Line boundaries identified using proposed Algorithm 4.2……	85
4.12a	Strips in printed Kannada script document…….....	86
4.12b	Line boundaries identified using proposed Algorithm 4.2 ……	86
4.13a	Strips in printed Tamil script document……..	86
4.13b	Line boundaries identified using proposed Algorithm 4.2…….	87
4.14a	Strips in printed Telugu script document……	87
4.14b	Line boundaries identified using proposed Algorithm 4.2…….	88
4.15a	Strips in printed Malayalam script document…….	88
4.15b	Line boundaries identified using proposed Algorithm 4.2…….	87
4.16	Urdu script document containing overlapping lines……...	91
4.17a	Printed text document of Gurmukhi script containing different font size text...	92
4.17b	Incorrect line boundaries of Figure 4.20a, identified using Algorithm 4.2……	93

4.18 Segmented lines, numbered 2 and 3, of Figure 4.20a using Algorithm 4.3…… 95
4.19 Word segmentation using vertical projection……. 96
4.20 Zone segmentation: (a) upper zone, (b) middle zone, (c) lower zone..... 96
4.21 Pronunciation of name, actual shape and example words of the vowels falling in first category..... 99
4.22 Pronunciation of name, actual shape and example words of the vowels containing sub-symbols in upper zone, falling in second category..... 100
4.23 Pronunciation of name, actual shape and example words of Gurmukhi characters having sub-symbols strokes in upper zone falling in third category. 100
4.24 Gurmukhi words containing touching sub-symbols in upper zone (touching sub-symbols have been marked with circles), (a) *bindī* touching with other sub-symbols, (b) *adhak* touching with other sub-symbols, (c) *tippī* touching with other sub-symbols……... 101
4.25 Words containing the touching characters in middle zone……. 102
4.26 Touching sub-symbols in lower zone (touching sub-symbols have been marked with circles), (a) lower zone sub-symbols touching with upper zone sub-symbols, (b) lower zone sub-symbols touching with each other, (c) two components of multi component vowel modifier ◌̱ touching with each other... 105
4.27 Segmentation process: (a) example word, (b) extended view of example word, (c) extended view of problem area, (d) top profile of problem area, (e) segmenting column in top profile, (f) actual segmented sub-symbols……... 109
4.28 Segmentation process in upper zone: (a) example words, (b) problem areas, (c) top profile of problem areas, (d) segmenting columns in top profiles, (e) actual segmented characters…… 110
4.29 Segmentation problems in upper zone: (a) example words, (b) problem areas, (c) top profile of problem areas, (d) incorrect segmentation for first and third words and no segmentation for second word, (e) segmented sub-symbols after implementing the Algorithm 4.4…….. 110
4.30 Touching sub-symbols in upper zone of Devanagari script: (a) touching sub-symbols which can be segmented using algorithm 4.4, (b) touching sub-symbols which can be segmented using some modification in algorithm 4.4,

(c) word containing three touching sub-symbols in upper zone.. 111
4.31a Horizontal and vertical projection of a word containing touching characters… 116
4.31b White dots showing start of headline, end of headline and possible locations of sidebar columns…….. 116
4.31c Segmented touching characters using case 1 of algorithm 4.5……... 116
4.32 Touching sub-symbols falling in fourth category (problem area encircled)….... 118
4.33 Problems in segmenting sub-symbols in middle zone……..... 119
4.34 Segmentation of touching sub-symbols of category c.1 in lower zone: (a) line containing touching sub-symbols of category c.1, (b) base line identified, (c) middle zone sub-symbols separated from lower zone, (d) lower zone sub-symbols separated from middle zone sub-symbols……. 120
4.35 Gurmukhi paragraph containing touching characters and heavily printed characters……..... 120
4.36a Part of Gurmukhi script document……..... 123
4.36b Segmented characters using standard character segmentation algorithms producing incorrect segmentation……... 124
4.36c Segmented characters using proposed algorithms…….. 124
5.1 Structural features of Gurmukhi characters: (a) sidebar present, (b) half sidebar present, (c) headline present, (d) two junctions with headline, (e) one junction with baseline …… 129
5.2 Degraded Gurmukhi characters having feature St4 false instead of true…….. 130
5.3 Projection profile of a sub-symbol…….. 131
5.4 Gurmukhi character *kakkā* (ਕ): (a) degraded character having loop, (b) degraded character having loop filled……..... 132
5.5 Projection profile of a Gurmukhi character…… 133
5.6 Example of WB encoding: (a) WB encoding for the white pixel at (6, 6), (b) WB encoding for the black pixel at (4, 6)……... 133
5.7 Determination of pixel density values for a particular window of the character matrix…….. 136
5.8 Percentage accuracy of DDD structural feature (St11) for sub-symbols in

	middle zone…….	149
5.9a	Percentage accuracy of transition feature for sub-symbols in middle zone for different values of T having M = 5…….....	149
5.9b	Percentage accuracy of transition feature for sub-symbols in middle zone for different values of T having M = 4…….....	150
5.9c	Percentage accuracy of transition feature for sub-symbols in middle zone for different values of T having M = 3…….....	150
5.10	Percentage accuracy of zoning feature for sub-symbols in middle zone for different grid sizes……..	151
5.11	Percentage accuracy of Zernike moments feature sub-symbols in middle zone for different order of Zernike moments……..	151
5.12	Percentage accuracy of OFM moments feature for sub-symbols in middle zone for different order of OFM moments…….	152
6.1	SFAM Neural network…….....	157
6.2a	A SFAM neural network before starting a learning process……..	159
6.2b	A SFAM neural network after the first input pattern has been learned……..	159
6.2c	A SFAM neural network after a number of learning steps…….	159
6.3	Separating hyperplane for feature selection where circles indicate the support vectors…….	161
6.4	Samples of degraded printed Gurmukhi characters for experimental purposes.	166
6.5	Gurmukhi script text image……	167
6.6	Gurmukhi script text image……	168
6.7	Gurmukhi script text image……	169
6.8	OCR output of the image in Figure 1.4……...	170
6.9	OCR output of the image in Figure 1.5……..	170
6.10	Recognition accuracy for different features using SFAM classifier……...	173
6.11a	Effect of different structural features on the recognition accuracy using k-NN classifier for different values of k ……..	176
6.11b	Effect of different statistical features on the recognition accuracy using k-NN classifier for different values of k……...	177
6.11c	Effect of all features on the recognition accuracy using k-NN classifier for	

different values of *k*……. 177
6.12 Recognition accuracy of different features using SVM classifier…….. 180
6.13 Comparison of recognition accuracy of different features using different classifiers……. 181

List of Tables

4.1 Percentage of occurrence of various types of strips in printed newspaper of Gurmukhi script …….. 71

4.2 Percentage of occurrence of strips containing two and more than two overlapping lines in newspaper documents ……..... 77

4.3 Types of various strips of Figures 4.5, 4.11a, 4.12a, 4.13a, 4.14a, 4.15a, 4.16a and 4.17a……. 78

4.4 Percentage of various types of strips in Devanagari, Bangla, Gujarati, Kannada, Tamil, Telugu and Malayalam scripts……. 78

4.5 Value of P1 and P2 for different scripts…….. 80

4.6 Sub-symbols of multicomponent characters and characters spanned to two zones in Gurmukhi script……. 97

4.7 Pronunciation of name, category and actual shape of vowels, auxiliary signs and other symbols containing sub-symbols in upper zone ……. 99

4.8 Properties of the touching sub-symbol categories in upper zone…… 101

4.9 Properties of the touching sub-symbol categories in middle zone..... 104

4.10 Properties of the touching sub-symbol categories in lower zone…… 106

4.11 Percentage accuracy in middle zone ……... 122

4.12 Percentage accuracy in upper zone…….. 122

5.1 Calculation of feature St7…… 131

5.2 Calculation of feature St11…….. 133

5.3 Set of different features used for different zones……..... 140

5.4 Feature values for Gurmukhi sub-symbols in middle zone for features St1 to St6…… 141

5.5 Left and right profile chain codes for Gurmukhi sub-symbols in middle zone... 142

5.6 Top and bottom profile chain codes for Gurmukhi sub-symbols in middle zone……. 143

5.7 Standard structural feature values for feature St6 for Gurmukhi sub-symbols in upper and lower zone……..... 144

5.8 Left and right profile chain codes for Gurmukhi sub-symbols in upper and

lower zone……... 145
5.9 Top and bottom profile chain codes for Gurmukhi sub-symbols in upper and lower zone……... 145
5.10 Analysis of structural features from St1 to St6……... 146
5.11 Percentage accuracy of structural feature St7 to St10 for middle zone sub-symbols…… 146
5.12 Percentage accuracy of structural feature DDD for middle zone sub-symbols... 147
5.13 Percentage accuracy of transitions feature for middle zone sub-symbols…….... 147
5.14 Percentage accuracy of zoning feature for middle zone sub-symbols…… 147
5.15 Percentage accuracy of Zernike moments feature for middle zone sub-symbols …….. 148
5.16 Percentage accuracy of OFMM feature for middle zone sub-symbols……... 148
5.17 Percentage accuracies of different features on Gurmukhi sub-symbols in upper and lower zone…….. 152
6.1 Recognition rate of Gurmukhi text images……..... 166
6.2 Recognition accuracy of structural features using SFAM neural network classifier……... 171
6.3 Recognition accuracy of statistical features using SFAM neural network classifier …….. 172
6.4 Recognition accuracy of all structural and statistical features using SFAM neural network classifier……. 172
6.5 Recognition accuracy of structural features using *k*-NN classifier for different values of *k*…… 174
6.6 Recognition accuracy of statistical features using *k*-NN classifier for different values of *k*…… 175
6.7 Recognition accuracy of all structural and statistical features using *k*-NN classifier for different values of *k*.……... 176
6.8 Recognition accuracy of structural features using SVM classifier..... 178
6.9 Recognition accuracy of statistical features using SVM classifier..... 179
6.10 Recognition accuracy of all structural and statistical features using SVM classifier…….. 179

6.11 Recognition accuracy of different features using different classifiers…… 180
6.12 Recognition accuracy using different classifiers on different sets of features for upper zone…….. 182
6.13 Recognition accuracy using different classifiers on different sets of features for lower zone…….. 183
6.14 Confusion matrix for Gurmukhi characters……. 184

Chapter 1
Introduction

The initial development of computers was for military and scientific purposes, where very few data had to be entered while a considerable amount of computations were to be done. On the other hand, in business applications, the amount of data to be entered is fairly high while the amount of computations to be performed is low. The problem of interchanging data between human beings and computing machines is a challenging one in business applications. Even today, the most commonly used method is the direct keyboard entry by an operator. This process is very slow, and the required human intervention can introduce errors and slows the process of data acquisition. As such, a natural solution would be to let the computers do the job for us: the machine would transform the original document into a more suitable form (in lesser time and having fewer errors) and process it. The presence of human operator would only be necessary if the system has problems with recognition, for correction purposes. At this point, the idea of Optical Character Recognition (OCR) evolved. The principal motivation for the development of OCR systems is the need to cope with the enormous flood of paper such as bank cheques, commercial forms, government records, credit card imprints, bill processing systems, airline ticket readers, passport readers and mail sorting systems generated by the expanding technological society.

In spite of the major efforts that have been expended to bring about a paper-free society, a very large number of paper-based documents are processed daily by computers all over the world in order to handle, retrieve and store information. The problem is that the manual process used to enter the data from these documents into computers demands a great deal of time and money. The field of Document Analysis and Recognition (DAR) has played a very important role in an attempt to overcome this problem. The general objective of DAR research is to fully automate the process of entering and understanding printed or handwritten data into the computer.

1.1 Historical Background

Initially in 1929, Tauschek obtained a patent on OCR in Germany, followed by Handel who obtained a US patent on OCR in USA in 1933 (U.S. Patent 1,915,993). Tauschek was also granted a US patent on his method in 1935 (U.S. Patent 2,026,329). Tauschek's machine was a mechanical device that used templates. A photodetector was placed so that when the template and the character to be recognised were lined up for an exact match, and a light was directed towards it, no light would reach the photodetector. The United States Postal Service has been using OCR machines to separate mail since 1965 based on engineering devised primarily by the productive inventor Jacob Rabinow.

The modern version of OCR is said to have originated in 1951 with David Shepard's invention: GISMO -A Robot Reader-Writer. This invention was closely followed by Jacob Rabinow's prototype machine in 1954, which was able to read upper case, printed output at the sluggish speed of one character per minute [1]. David Shepard's company Intelligent Machines Research (IMR) is said to have been the first to apply optical reading techniques in a commercial situation. They installed a system at Reader's Digest in 1955 [2]. A number of other companies, including IBM, were conducting research into OCR throughout the early 60's, culminating in the first marketable commercial OCR system: IBM's 1418 [3]. Early systems, such as the one mentioned above, were very constrained in the sense that they were bound to read special, artificial fonts. These types of OCR systems are usually associated with the first generation of OCRs. The second generation, on the other hand, was characterized by hand-printed character recognition capabilities [3]. In the early stages, only numerals could be recognised by these pioneering machines. One such device was IBM's 1287 OCR system. This was the first of the second-generation machines, and one of the most famous ones. It was originally exhibited at the World's Fair in 1965 [3]. It has been approximately 40 years since the first automatic handwriting readers were proposed. Since then, incredible progress has been made to enable computers to recognise, interpret and identify machine-printed and handwritten text. Extensive research has been carried out in terms of technical papers and reports by various researchers around the world.

Pal and Chaudhuri [4] have divided the commercial OCR system into four generations, depending on versatility, robustness and efficiency. First generation includes commercialized

OCR, IBM 1418, while the second generation includes famous OCR, IBM 1287. Third generation has OCRs for hand-printed and poor print quality characters developed during 1975-1985 [4-6]. Fourth generation has OCRs of documents with text, graphics, tables and mathematical symbols *etc.* The OCRs of this generation generally have an accuracy of less than 85%. Research on complex documents is in progress [7]. Research into handwriting recognition still continues to be intense during all these years. The motivation may be attributed to the challenging nature of the character recognition problem and the countless number of commercial applications that it may be applied to [8].

1.2 Character Recognition

OCR is a process which associates a symbolic meaning with objects (letters, symbols and numbers) drawn on an image, *i.e.,* OCR techniques associate a symbolic identity with the image of a character. OCR can also be defined as the process of converting scanned images of machine-printed or handwritten text (numerals, letters, and symbols), into a computer processable format. The ultimate goal of an OCR is to imitate the human ability to read at a much faster rate by associating symbolic identities with images of characters. The practical importance of OCR applications, as well as the interesting nature of the OCR problems has led to great research interest and measurable advances in this field.

The field of Document Analysis and Recognition is vast and it contains many applications. Character recognition is one of the branches of DAR. As shown in Figure 1.1, the problem of character recognition can be divided into printed and handwritten character recognition. Handwritten character recognition has been further divided into off-line and on-line handwritten character recognition [7]. Off-line handwriting recognition refers to the process of recognising words that have been scanned from a surface (such as a sheet of paper) and are stored digitally in grey scale format. After being stored, it is conventional to perform further processing to allow superior recognition. In the on-line case, the handwriting is captured and stored in digital form via different means. Usually, a special pen is used in conjunction with an electronic surface. As the pen moves across the surface, the two-dimensional coordinates of successive points are represented as a function of time and are stored in order [7]. It is generally accepted that the on-line method of recognising

handwritten text has achieved better results than its off-line counterpart. This may be attributed to the fact that more information may be captured in the on-line case such as the direction, speed and the order of strokes of the handwriting.

On the other side machine-printed character recognition can be on good quality documents or degraded printed documents. The degraded documents can be of many types having different kinds of problems. We have discussed in detail various kinds of degradations and the problems associated with them in printed documents in Chapter 2.

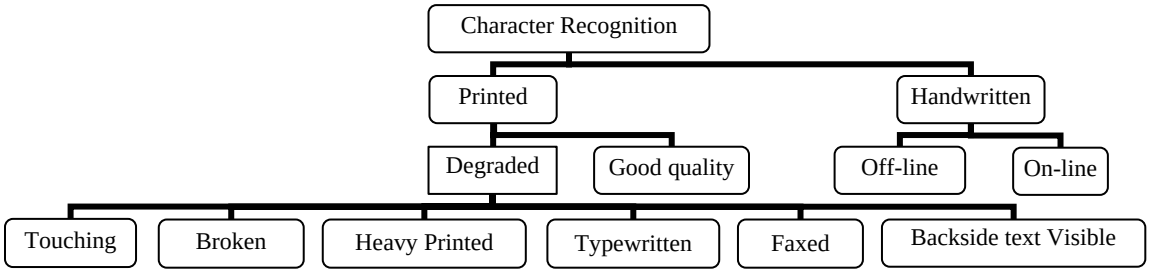


Figure 1.1: Hierarchy of character recognition problems

1.3 Components of an OCR

The process of an OCR of any script can be divided into phases as shown in Figure 1.2. Each phase has been explained below:

1.3.1 Digitization

It refers to the process of converting a paper- or film-based document into electronic form. The electronic conversion is accomplished through imaging a process whereby a document is scanned and an electronic representation of the original, in the form of a bitmap image, is produced. The imaging process involves recording changes in light intensity reflected from the document as a matrix of dots. The light/color value(s) of each dot is stored in binary digits. One bit would be required for each dot in a binary scan, whereas up to 32 bits could be required per dot for a color scan. Digitization produces the digital image, which is fed to the pre-processing phase.

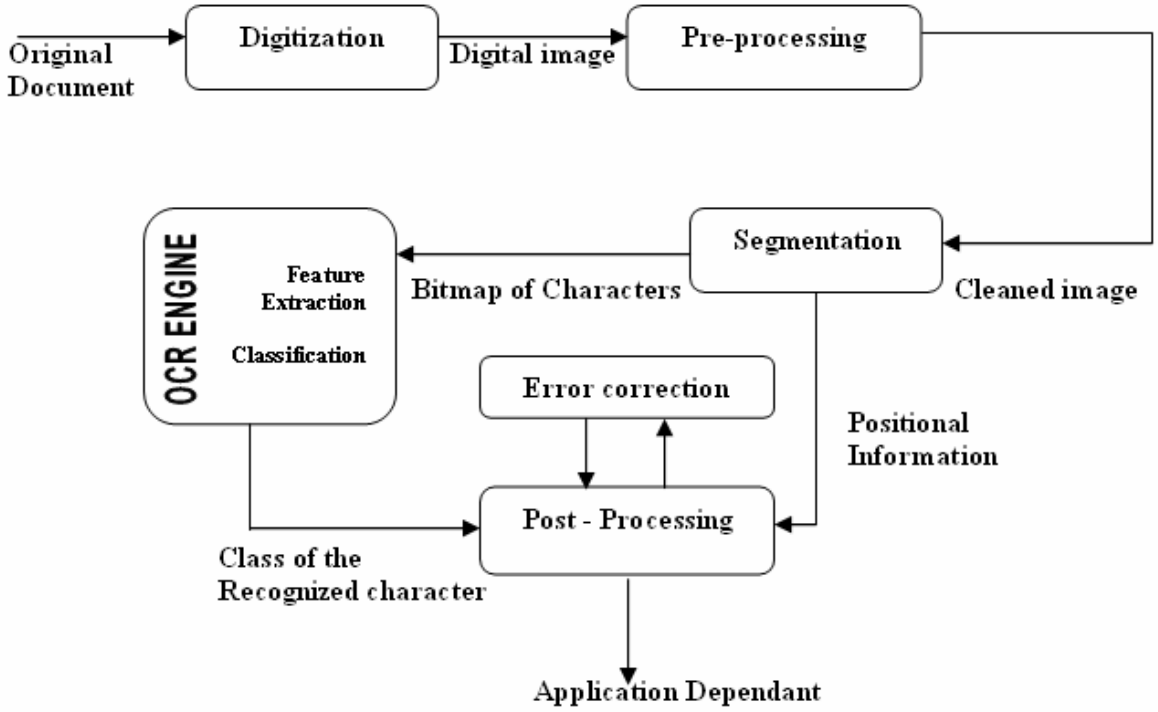


Figure 1.2: Block diagram for a typical OCR application

1.3.2 Pre-processing

Pre-processing is the name given to a family of procedures for smoothing, enhancing, filtering, cleaning-up and otherwise massaging a digital image so that subsequent algorithms along the road to final classification can be made simple and more accurate. Pre-processing methods include noise removal, smoothing, skew correction and skeletonization (also called thinning). The major objective of *noise removal* is to remove any unwanted bit-patterns, which do not have any significance in the output. The objective of *contour smoothing* is to smooth contours of broken and/or noisy input characters. *Skewness* refers to the tilt in the bit-mapped image of the scanned paper for OCR. It is usually caused if the paper is not fed straight into the scanner. Most of the OCR algorithms are sensitive to the orientation (or skew) of the input document image, making it necessary to develop algorithms which can detect and correct the skew automatically. *Skeletonization* refers to the process of reducing the width of a line like object from many pixels wide to just single pixel. This process can remove irregularities in letters and in turn, makes the recognition algorithm simpler because

they only have to operate on a character stroke, which is only one pixel wide. It also reduces the memory space required for storing the information about the input characters and no doubt, this process reduces the processing time too. After pre-processing phase, we have a cleaned image that goes to the segmentation phase.

1.3.3 Segmentation

It is an operation that seeks to decompose an image of sequence of characters into sub-images of individual symbols. Character segmentation is a key requirement that determines the utility of conventional OCR systems. It includes line, word and character segmentation. Different methods used can be classified based on the type of text and strategy being followed like straight segmentation method, recognition-based segmentation and cut classification method.

1.3.4 Feature extraction

As shown in Figure 1.2, the OCR engine consists of two stages, feature extraction and classification. Feature extraction is the name given to a family of procedures for measuring the relevant shape information contained in a pattern so that the task of classifying the pattern is made easy by a formal procedure. The feature extraction stage analyses a text segment and selects a set of features that can be used to uniquely identify the text segment. The selection of a stable and representative set of features is the heart of pattern recognition system design. Among the different design issues involved in building an OCR system, perhaps the most significant one is the selection of the type and set of features.

1.3.5 Classification

The second step of OCR engine, classification stage, is the main decision making stage of an OCR system and uses the features extracted in the previous stage to identify the text segment according to preset rules. Classification is concerned with making decisions concerning the class membership of a pattern in question. The task in any given situation is to design a

decision rule that is easy to compute and will minimize the probability of misclassification relative to the power of feature extraction scheme employed. If we assume that *d* features are observed on a pattern or object, then we can represent the pattern by a *d*-dimensional vector $X = (x_1, x_2, ..., x_d)$ and usually refer to X as a *feature vector* and the space in which X lies as the *feature space*. Patterns are thus transformed by feature extraction process into points in *d*-dimensional feature space. A pattern class can then be represented by a region or sub-space of the feature space. Classification then becomes a problem of determining the region of feature space in which an unknown pattern falls.

1.3.6 Post-processing

OCR results usually contain errors because of character classification and segmentation problems. For the correction of recognition errors, OCR systems apply contextual post-processing techniques. The two most common post-processing techniques for *error correction* are dictionary lookup and statistical approach. The advantage of statistical approach over dictionary-based methods is computational time and memory utilization. Conversely, lexical knowledge about entire words is more accurate when using a dictionary.

1.4 Need for Study

These days, PC-based systems are commercially available to read good quality documents of a single font with very high accuracy and documents of multiple fonts with reasonable accuracy. However, most of the available work is on European, Chinese and Japanese scripts. A few reports are available on Indian script recognition systems of fine quality printed text and only a few deals with a complete OCR for printed documents. Out of the few papers, which were found in literature dealing with complete computer recognition of an Indian language script, one is by Pal and Chaudhuri [9, 10]. They have made OCRs on Bangla and Devanagari scripts. Another is by Lehal and Singh [11]. Authors have developed an OCR of Gurmukhi script for good quality Gurmukhi text. But the work done by these authors has some limitations, such as it can only recognise good quality documents. The authors have not discussed in detail any solution for the recognition of degraded documents containing

touching characters and heavily printed (self touching) characters. As reported by Bosker [12], the accuracy of an OCR system decreases when input documents contain touching characters. Hence, it is the need of the hour to deal with the problems due to degradation of documents during recognition process.

In recent times, the problem of machine-printed character recognition has been significantly solved. Many commercial and accurate systems are now available [1]. Unfortunately, the success obtained with the machine-printed OCR systems has not readily been transferred to the handwriting recognition arena and degraded printed document recognition.

The reported work in the field of developing a PC-based system for recognising the degraded text is very less. There is no work done on this topic in an Indian script, in general, and in Gurmukhi script, in particular. The present work is the first attempt towards the complete computer recognition of degraded Gurmukhi script. This work will also facilitate the development of such systems for recognition of degraded text for other Indian scripts that are structurally similar to Gurmukhi script.

1.5 Motivation

Paper documents are ubiquitous in our society, but there is still a need to have many of these documents in electronic form. There are large collections of data originating in paper form, such as office data, books, and historical manuscripts that would benefit by being available in digital form. Once converted to electronic form, the documents can be used in digital libraries or for wider dissemination on the internet. To increase efficiency of paperwork processing and to better utilize the content of data originating in paper form, it is required that the paper documents be converted into images of the documents in electronic form, and that must be converted to a computer-readable format such as ASCII format. This will allow editing as well as search and retrieval of information on these documents. The conversion is done by scanning the document and then processing the digital image with an OCR algorithm.

The operations of scanning, printing, photocopying, and faxing introduce degradations to the images. Human beings often notice the degradations present in photocopies of

photocopies of text documents. Smaller image degradations, which may not be noticeable to humans, are still large enough to interfere with the ability of a computer to read printed text.

These degradations can lead to broken or touching characters, which are the major causes of OCR errors. As shown in Figure 1.3(b), a negative edge displacement will result in thinning of strokes which can cause an “m” to resemble with the pair of letters “r n” or for an “e” to resemble with a “c”. If the edge displacement is positive, the stroke width will increase, which can cause an “r n” to resemble with an “m” as shown in Figure 1.3(c).

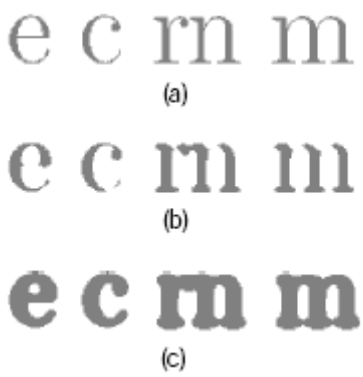
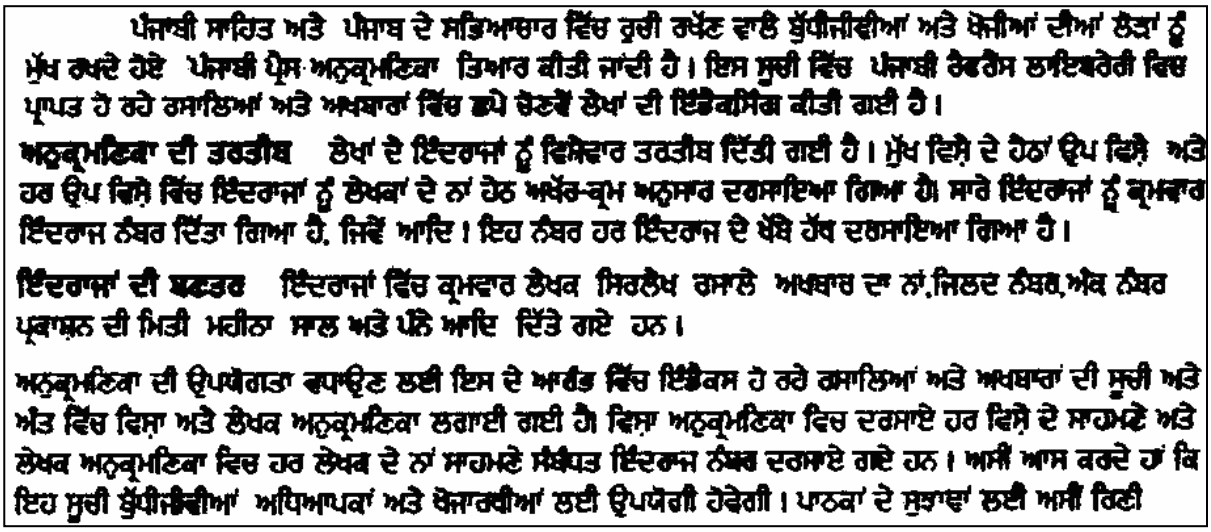


Figure 1.3: Examples of characters with different degradations that lead to common OCR errors: (a) true characters, (b) characters with negative edge displacement, (c) characters with positive edge displacement

Despite four decades of extensive research, significant progress has only been made in limited areas such as developing algorithms for isolated machine-printed characters. OCR systems for document recognition still have limited capabilities for recognising degraded documents [13, 14]. Research and experiments have shown that the performance breakdown of commercial document recognition system under real application situation is caused by the difficulty in dealing with touching characters and heavily printed characters that are abundant in real-world documents as a result of document degradations [6, 14-16]. Figures 1.4 and 1.5 contain the Gurmukhi script documents containing touching characters and heavily printed characters. An OCR system developed for recognition of good quality Gurmukhi characters by Lehal and Singh [10] performs only with an accuracy of less than 30% on these documents. Therefore, it is required to solve the problem of touching characters and heavily printed characters to improve the recognition accuracy. Touching characters make it difficult to correctly segment character images for individual classification, and therefore pose severe

difficulty to conventional document recognition system that is critically dependent on character segmentation.



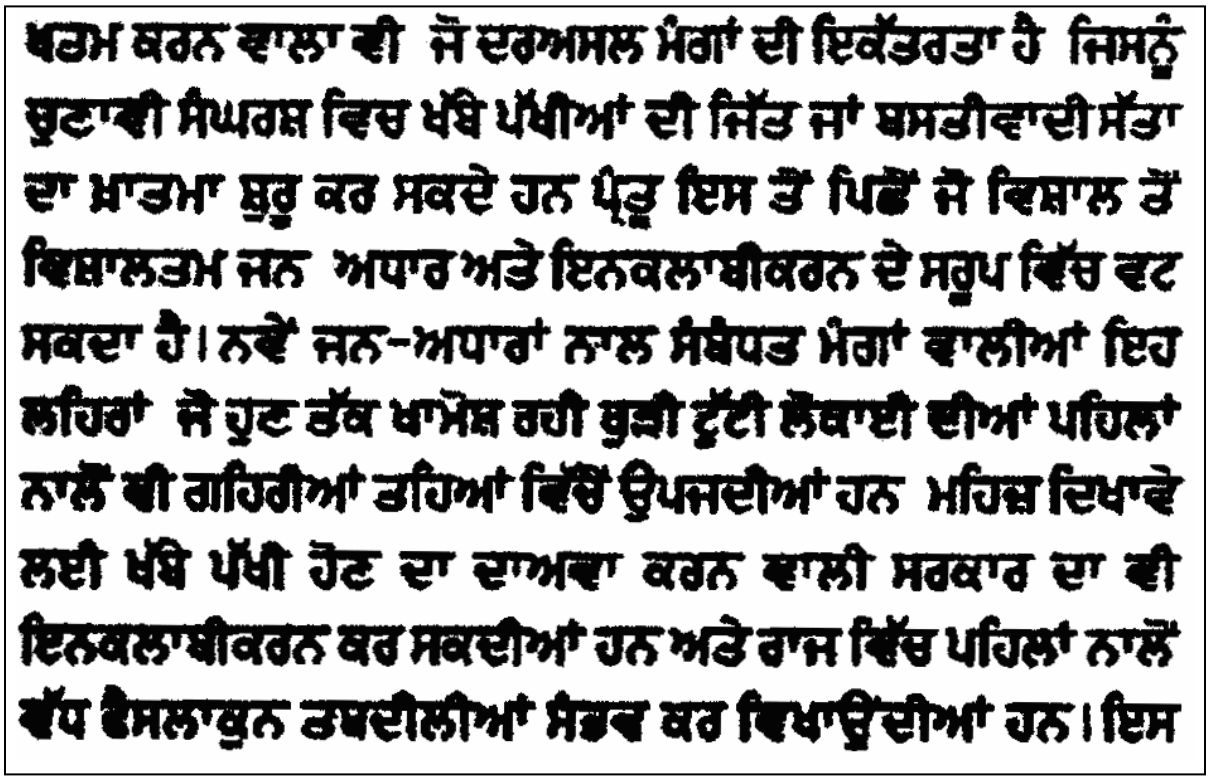
ਪੰਜਾਬੀ ਸਾਹਿਤ ਅਤੇ ਪੰਜਾਬ ਦੇ ਸੱਭਿਆਚਾਰ ਵਿੱਚ ਰੁਚੀ ਰਖੱਣ ਵਾਲੇ ਬੁੱਧੀਜੀਵੀਆਂ ਅਤੇ ਖੋਜੀਆਂ ਦੀਆਂ ਲੋੜਾਂ ਨੂੰ ਮੁੱਖ ਰਖਦੇ ਹੋਏ ਪੰਜਾਬੀ ਪ੍ਰੈਸ ਅਨੁਕ੍ਰਮਣਿਕਾ ਤਿਆਰ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਸ ਸੂਚੀ ਵਿੱਚ ਪੰਜਾਬੀ ਰੈਫਰੈਂਸ ਲਾਇਬਰੇਰੀ ਵਿਚ ਪ੍ਰਾਪਤ ਹੋ ਰਹੇ ਰਸਾਲਿਆਂ ਅਤੇ ਅਖਬਾਰਾਂ ਵਿੱਚ ਛਪੇ ਚੋਣਵੇਂ ਲੇਖਾਂ ਦੀ ਇੰਡੈਕਸਿੰਗ ਕੀਤੀ ਗਈ ਹੈ।

ਅਨੁਕ੍ਰਮਣਿਕਾ ਦੀ ਤਰਤੀਬ ਲੇਖਾਂ ਦੇ ਇੰਦਰਾਜਾਂ ਨੂੰ ਵਿਸ਼ੇਵਾਰ ਤਰਤੀਬ ਦਿੱਤੀ ਗਈ ਹੈ। ਮੁੱਖ ਵਿਸ਼ੇ ਦੇ ਹੇਠਾਂ ਉਪ ਵਿਸ਼ੇ ਅਤੇ ਹਰ ਉਪ ਵਿਸ਼ੇ ਵਿੱਚ ਇੰਦਰਾਜਾਂ ਨੂੰ ਲੇਖਕਾਂ ਦੇ ਨਾਂ ਹੇਠ ਅੱਖਰ-ਕ੍ਰਮ ਅਨੁਸਾਰ ਦਰਸਾਇਆ ਗਿਆ ਹੈ। ਸਾਰੇ ਇੰਦਰਾਜਾਂ ਨੂੰ ਕ੍ਰਮਵਾਰ ਇੰਦਰਾਜ ਨੰਬਰ ਦਿੱਤਾ ਗਿਆ ਹੈ, ਜਿਵੇਂ ਆਦਿ। ਇਹ ਨੰਬਰ ਹਰ ਇੰਦਰਾਜ ਦੇ ਖੱਬੇ ਹੱਥ ਦਰਸਾਇਆ ਗਿਆ ਹੈ।

ਇੰਦਰਾਜਾਂ ਦੀ ਬਣਤਰ ਇੰਦਰਾਜਾਂ ਵਿੱਚ ਕ੍ਰਮਵਾਰ ਲੇਖਕ ਸਿਰਲੇਖ ਰਸਾਲੇ ਅਖਬਾਰ ਦਾ ਨਾਂ,ਜਿਲਦ ਨੰਬਰ,ਅੰਕ ਨੰਬਰ ਪ੍ਰਕਾਸ਼ਨ ਦੀ ਮਿਤੀ ਮਹੀਨਾ ਸਾਲ ਅਤੇ ਪੰਨੇ ਆਦਿ ਦਿੱਤੇ ਗਏ ਹਨ।

ਅਨੁਕ੍ਰਮਣਿਕਾ ਦੀ ਉਪਯੋਗਤਾ ਵਧਾਉਣ ਲਈ ਇਸ ਦੇ ਆਰੰਭ ਵਿੱਚ ਇੰਡੈਕਸ ਹੋ ਰਹੇ ਰਸਾਲਿਆਂ ਅਤੇ ਅਖਬਾਰਾਂ ਦੀ ਸੂਚੀ ਅਤੇ ਅੰਤ ਵਿੱਚ ਵਿਸ਼ਾ ਅਤੇ ਲੇਖਕ ਅਨੁਕ੍ਰਮਣਿਕਾ ਲਗਾਈ ਗਈ ਹੈ। ਵਿਸ਼ਾ ਅਨੁਕ੍ਰਮਣਿਕਾ ਵਿਚ ਦਰਸਾਏ ਹਰ ਵਿਸ਼ੇ ਦੇ ਸਾਹਮਣੇ ਅਤੇ ਲੇਖਕ ਅਨੁਕ੍ਰਮਣਿਕਾ ਵਿਚ ਹਰ ਲੇਖਕ ਦੇ ਨਾਂ ਸਾਹਮਣੇ ਸੰਬੰਧਤ ਇੰਦਰਾਜ ਨੰਬਰ ਦਰਸਾਏ ਗਏ ਹਨ। ਅਸੀਂ ਆਸ ਕਰਦੇ ਹਾਂ ਕਿ ਇਹ ਸੂਚੀ ਬੁੱਧੀਜੀਵੀਆਂ ਅਧਿਆਪਕਾਂ ਅਤੇ ਖੋਜਾਰਥੀਆਂ ਲਈ ਉਪਯੋਗੀ ਹੋਵੇਗੀ। ਪਾਠਕਾਂ ਦੇ ਸੁਝਾਵਾਂ ਲਈ ਅਸੀਂ ਰਿਣੀ

Figure 1.4: Degraded printed Gurmukhi script document containing touching characters and heavily printed characters



ਖਤਮ ਕਰਨ ਵਾਲਾ ਵੀ ਜੋ ਦਰਅਸਲ ਮੰਗਾਂ ਦੀ ਇਕੱਤਰਤਾ ਹੈ ਜਿਸਨੂੰ ਚੁਣਾਵੀ ਸੰਘਰਸ਼ ਵਿਚ ਖੱਬੇ ਪੱਖੀਆਂ ਦੀ ਜਿੱਤ ਜਾਂ ਬਸਤੀਵਾਦੀ ਸੱਤਾ ਦਾ ਖਾਤਮਾ ਸ਼ੁਰੂ ਕਰ ਸਕਦੇ ਹਨ ਪ੍ਰੰਤੂ ਇਸ ਤੋਂ ਪਿਛੋਂ ਜੋ ਵਿਸ਼ਾਲ ਤੋਂ ਵਿਸ਼ਾਲਤਮ ਜਨ ਅਧਾਰ ਅਤੇ ਇਨਕਲਾਬੀਕਰਨ ਦੇ ਸਰੂਪ ਵਿੱਚ ਵਟ ਸਕਦਾ ਹੈ। ਨਵੇਂ ਜਨ-ਅਧਾਰਾਂ ਨਾਲ ਸੰਬੰਧਤ ਮੰਗਾਂ ਵਾਲੀਆਂ ਇਹ ਲਹਿਰਾਂ ਜੋ ਹੁਣ ਤੱਕ ਖਾਮੋਸ਼ ਰਹੀ ਬੁੜੀ ਟੁੱਟੀ ਲੋਕਾਈ ਦੀਆਂ ਪਹਿਲਾਂ ਨਾਲੋਂ ਵੀ ਗਹਿਰੀਆਂ ਤਹਿਆਂ ਵਿੱਚੋਂ ਉਪਜਦੀਆਂ ਹਨ ਮਹਿਜ਼ ਦਿਖਾਵੇ ਲਈ ਖੱਬੇ ਪੱਖੀ ਹੋਣ ਦਾ ਦਾਅਵਾ ਕਰਨ ਵਾਲੀ ਸਰਕਾਰ ਦਾ ਵੀ ਇਨਕਲਾਬੀਕਰਨ ਕਰ ਸਕਦੀਆਂ ਹਨ ਅਤੇ ਰਾਜ ਵਿੱਚ ਪਹਿਲਾਂ ਨਾਲੋਂ ਵੱਧ ਫੈਸਲਾਕੁਨ ਤਬਦੀਲੀਆਂ ਸੰਭਵ ਕਰ ਵਿਖਾਉਂਦੀਆਂ ਹਨ। ਇਸ

Figure 1.5: Degraded printed Gurmukhi script document containing touching characters and heavily printed characters

The objective of this research is to propose new approaches to recognition of degraded documents containing touching characters and heavily printed characters only. OCR algorithms can achieve good recognition rates (near 99%) on images with little degradation. However, recognition rates drop to 70% or lower when image degradations are present. Typical pages of text have more than 2000 characters per page. Therefore, an error rate of 30% results in more than 600 mistakes. Before the mistakes can be corrected, they must be located, making the correction process even more tedious.

1.6 Objectives of this Research

The objectives of the proposed study were outlined as follows:

1. To develop algorithms and data structures for OCR of degraded printed Gurmukhi script.
2. To develop software, based on the above algorithms, which will convert any scanned and pre-processed document of degraded printed Gurmukhi script into machine-readable form.

In order to accomplish these objectives, a comprehensive study of various methods used in the development of OCR systems for different Indian and other scripts has been carried out. This thesis describes a complete OCR system for recognising degraded printed Gurmukhi script documents. Characteristics of Gurmukhi script and other Indian scripts have been studied for the complementation. New algorithms have been designed for segmentation of touching characters, as the existing algorithms were not found suitable for segmentation of these characters. More robust features have been selected during feature extraction phase.

1.7 Major Contributions and Achievements

The main contributions of this thesis can be summarised as follows.

1. A detailed literature survey of different phases of OCR has been done. Also literature survey of Indian script recognition and degraded text recognition has been done. Different kinds of degradations in degraded printed Gurmukhi script have been identified. The problems associated with each kind of degradation, the sources of degradation, some possible solutions and comparison of each kind of degradation with corresponding degradation in Roman script has been made.
2. New algorithms have been proposed for segmenting the horizontally overlapping lines and associating the broken components of a line to their respective lines in printed Gurmukhi script. The same algorithms have been tested on Devanagari, Bangla, Gujarati, Kannada, Tamil, Telugu and Malayalam scripts and 95-99.7% accuracy has been obtained for the purpose depending upon the script.
3. New algorithms based on the structural properties of the Gurmukhi script have been proposed for segmenting touching characters in degraded printed Gurmukhi script. Separate algorithms have been proposed for segmenting touching character in upper, middle and lower zones of Gurmukhi script.
4. Features have been selected which are more robust to touching characters and heavily printed characters. Structural and statistical features have been used for making a feature vector for recognition purpose.
5. Various classifiers such as *k*-Nearest Neighbor (*k*-NN), Support Vector Machines (SVM) and Artificial Neural Network (ANN) have been used for the recognition purpose. These classifiers have been used for the recognition of the individual characters in all three zones (upper, middle and lower) of the degraded printed Gurmukhi script text.

Currently, there is no software available for OCR of degraded Gurmukhi script, in particular, and other degraded Indian language scripts, in general. There is a dire need for the OCRs of Indian language scripts as people working in Indian language scripts are denied the opportunity of converting scanned images of degraded machine-printed or handwritten text into a computer processable format. This work is the first attempt towards the development of an OCR for degraded printed Gurmukhi script and it can lead towards the development of

OCRs for recognising degraded documents of other Indian language scripts such as Devanagari and Bangla that are structurally similar to Gurmukhi script.

1.8 Assumptions

We have considered following assumptions while developing algorithms and performing experiments in this thesis:

1. We have considered documents whose binarization (digitization) has already been done.
2. Noise cleaning, orientation and skew detection and corrections have already been done. We have considered pre-processed documents only.
3. Text-non text separation has already been done and we are considering only single column printed text documents.
4. Main types of degradations discussed in the thesis are overlapping lines (inter-line overlap and touching), touching characters (inter-character overlap and touching), heavy printed characters (character bleeding/thickening).

1.9 Organization of Thesis

Rest of this thesis is organized as follows. In Chapter 2, we have presented review of literature. The review is organized according to different stages of an OCR system. Besides this, review of Indian script recognition and degraded text recognition has been done in this chapter. Chapter 3 includes discussion about different kinds of degradations that are found in printed Gurmukhi script. In Chapter 4, we have proposed new algorithms for segmenting text into lines, lines into words and subsequently words into characters. Chapter 5 presents various feature extraction methods used for extracting features of degraded printed Gurmukhi characters. We have here used structural and statistical methods for extracting features of Gurmukhi characters. Chapter 6 contains discussion on various classifiers used for recognition purpose. Finally, in Chapter 7, thesis is concluded with a summary and discussion on future research directions.

Chapter 2

Review of Literature

OCR is one of the oldest ideas in the history of pattern recognition using computers. In spite of its age, the state of art in the field could reach the point of practical usage in recent times only. The process of OCR starts with reading of a scanned image of a series of characters, determines their meaning, and finally translates the image to a computer written text document. It has commonly been used by the post-offices to mechanically read names and addresses on envelopes and by the banks to read amount and number on cheques. Also, companies and civilians can use this method to quickly translate paper documents to computer written documents. As of now, there exist several commercial OCR softwares and there has been quite a lot of research in the area.

It has been found that a neural network based multiresolution OCR delivered more than about 99.6% recognition rate on the isolated printed multifont characters in Roman script [1]. However, it remains a challenge till now to develop an OCR system which can achieve such a high recognition rate, regardless of the quality of the input document.

A lot of research has been done on OCR in last 55 years. Some books [17-19] and many surveys [3, 4, 20-38] have been published on the character recognition. Most of the published work on OCR has been on Latin characters, with work on Japanese and Chinese characters emerging in the middle of 1960s. Useful reviews and surveys in the field of OCR include the historical review of OCR methods and commercial systems by Mori [3], Mantas [20], Govindan and Shivaprasad [21] and Suen *et al.* [22]. The survey by Impedovo *et al*. [23] focuses on commercial OCR systems, while the work by Tian *et al*. [24] surveys the area of machine-printed OCR. Jain *et al.* [25] summarised and compared some of the well-known methods used in various stages of a pattern recognition system. They have tried to identify research topics and applications which are at the forefront in this field. Pal and Chaudhuri [4] in their survey report summarised different systems for Indian language scripts recognition.

They have described some commercial systems like Bangla and Devanagari OCRs. They reported the scope of future work to be extended in several directions such as OCR for poor quality documents, for multi font OCR and bi-script/multi-script OC*R* development *etc.* A bibliography of the fields of OCR and document analysis is given in [26]. Tappet *et al*. [27] and Wakahara *et al*. [28] surveyed on-line handwriting recognition and described a distortion-tolerant shape matching method. Noubound and Plamondon [29] and Suen *et al*. [22] proposed a survey on methods used for on-line recognition of hand-printed characters while Connell *et al*. [30, 31] described on-line character recognition for Devanagari characters and alphanumeric characters.

Bortolozzi *et al.* [32] have published a very useful study on recent advances in handwriting recognition. Lee *et al*. [33] described off-line recognition of totally unconstrained handwritten numerals using multiplayer cluster neural network. The character regions are determined by using projection profiles and topographic features extracted from the gray-scale images. Then, a nonlinear character segmentation path in each character region is found by using multi-stage graph search algorithm.

Khaly and Ahmed [34], Amin [35] and Lorigo & Govindraju [36] have produced a comprehensive survey and bibliography of research on the Arabic optical text recognition. Hildebrandt and Liu [37] have reported the advances in handwritten Chinese character recognition and Liu *et al.* [38] have discussed various techniques used for on-line Chinese character recognition.

2.1 Segmentation

In document analysis, segmentation is the synonym for line, word or character segmentation. According to Casey and Lecolinet [39], segmentation points should not solely be created on the basis of local information. Rather, previous and future segmentation decisions should also be made on the basis of contextual information. In other words, if segmentation is made between two primitives in a word and the resulting letters do not fit with any letter permutations in a word lexicon, such segmentation may be deemed incorrect and may influence previous and future segmentations. Casey and Lecolinet [39] summarises the last forty years of character segmentation research as being dependent on local topological

information as well as global contextual information. A few surveys have been published in the literature on the specific topic of character segmentation [39-42]. Dunn and Wang [40] have divided the segmentation techniques reviewed in their paper into straight segmentation and segmentation-recognition categories. Lu and Shridhar [42] have studied the segmentation of hand-printed words, handwritten numerals and cursive handwritten words. Casey and Lecolinet [39] have divided their survey into four categories: dissection techniques, recognition-based segmentation, mixed strategies (oversegmentation) and holistic strategies as discussed below:

(a) Dissection techniques: These refer to techniques for segmentation that are based on the concept of segmenting character images into sub-components utilizing general features. No classification, contextual knowledge or character shape discrimination steps are present in these segmentation techniques.

(b) Recognition-based segmentation: These techniques do not employ specific dissection strategies. Rather the image is partitioned into overlapping sections and a classifier is used to perform segmentation by verifying whether a particular section consists of a character. This type of technique is referred to as recognition-based because the character segmentation is a by-product of recognition.

(c) Hybrid strategies (over-segmentation): These strategies are the combination of the first two that are mentioned above. Dissection is used to over-segment the word or connected character component into a sufficient number of components as to encompass all segmentation boundaries present. In the next step, classification is employed to determine the optimum segmentations from a set of possible segmentation hypotheses.

(d) Holistic strategies: Finally, these strategies intend to recognise words as entire units rather than attempting to extract individual characters.

Initial systems for segmenting machine-printed characters were based on two simple features: white space and pitch [39]. White space refers to gaps between printed characters

that could be detected by vertically scanning the image. A column in the image that did not contain any black pixels (foreground pixels) could be considered a white space. The pitch relates to machine print applications that output characters of fixed width. It is the number of characters that occupy a given area in the horizontal direction. Pitch information could therefore be used to determine the location of equally spaced segmentation points in a line of printed information. This knowledge could be effective in splitting blurred or merged character boundaries. Hoffman and McCullough [43] also utilized pitch estimation of printed characters along with an evaluation function based on a count of black-white and white-black transitions to estimate segmentation regions.

Another popular method used in detecting segmentation points is that of projection analysis [39]. In particular, vertical projection or vertical histograms are used to tally the number of black pixels in each vertical column in the printed image. It has served such purposes as the detection of white spaces or areas of low pixel density in printed matter. It can also be used to estimate the presence of vertical strokes in machine print. Baird *et al*. [44] utilized the vertical projection to determine segmentation zones in overlapping or touching characters. They calculated the ratio of the histogram curve's second derivative to its height. Minima in the histogram would appear as peaks following ratio calculation and hence would indicate prospective splitting points. Lu [41], Tsujimoto and Asada [45] and Liang *et al*. [46] proposed other methods, based on the vertical histogram. Wang and Jean [47] proposed a method to segment touching characters using neural networks. These were built on the research presented by Baird *et al*. [44]. All these segmentation techniques belong to the Dissection category without a recognition module to confirm or reinforce character splitting.

Using recognition-based segmentation technique, Casey and Nagy [48] developed a recursive splitting algorithm that utilizes a window to scan a printed image from left to right, testing all segmentation possibilities. To begin with, a window is placed over the entire input image. The window is then gradually narrowed from the right. At each stage of narrowing, the prototype classifier attempts to recognise the contents of the window. This continues until the recogniser makes a match with a character or the window becomes too small. If a character is successfully recognised, the window is moved left up to the point of truncation and the window is reset to once again begin narrowing from the extreme right.

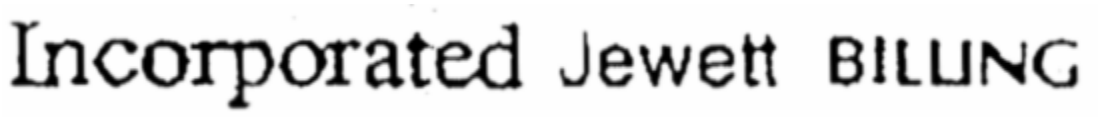
One of the most commonly found problems in degraded machine-printed documents is existence of touching characters. When two neighboring characters touch each other, they are called merged characters or touching characters. Segmentation of touching characters is an important and difficult task. In the following sub-sections, we have reviewed various techniques used in literature for segmenting touching characters/numerals.

2.1.1 Segmentation of touching characters

Segmentation of touching characters (sometimes referred to as composite characters or merged characters) is a difficult problem in character segmentation. There are two key issues involved in this problem, the first is to determine which segments contain multiple characters, *i.e.*, identifying the candidate of segmentation, and the second is to find break locations in candidate of segmentation for segmenting touching characters [48].

The techniques for segmenting the merged characters can be divided into two categories [41], feature based and recognition based. In the feature based technique, vertical projection is transformed to a function which provides more direct information for finding the break points. The second technique involves segmentation and recognition process carried simultaneously for segmenting and recognising touching characters.

The most commonly used feature for detecting multiple character segments (candidate of segmentation) is the width and the aspect ratio of the character. Lu [41] suggested that character width can be dynamically estimated during the segmentation process. Each candidate segment is then examined by comparing its width with the estimated character width or by measuring the aspect ratio of the segment. It is well understood that most characters have widths smaller than their height and a single character segment should have a width of less than twice of the estimated character width. The combination of these two measurements works well most of the time but fails in special cases. Figure 2.1.(a) shows examples of touching character segments, “In”, “rp”, “ed”, that are wider than single characters in the image, and so they can be identified using either the segment width or the aspect ratio.



(a) (b) (c)

Figure 2.1: Touching characters in Roman script (Dunn *et al.* [40])

However, in Figure 2.1(b) touching character segment “tt” is narrower than “J” and “w”, and touching character segment “LI” is no wider than “N” and “G” in the Figure 2.1(c). Furthermore, the aspect ratios of “tt” and “LI” are as normal as a single character segment. Lu [41] further proposed generating single and multiple character profile models for discriminating multi character segments from single character segments.

After finding the candidate of segmentation, the next issue is to find the break location which will segment touching characters. It is a splitting process to determine where to split the multiple characters region. Casey and Lecolinet [39] have discussed a few techniques for finding these break locations in their paper. Kahan *et al.* [52] proposed an objective function for finding breaking points within the merged characters. The function is the ratio of the second difference of the vertical projection function $V(x)$ to $V(x+1)$, namely,

$$f(x-1, x, x+1) = \frac{V(x-1) - 2 \times V(x) + V(x+1)}{V(x)} \tag{2.1}$$

The maxima of this objective function were used as the possible breakpoints. Liang *et al*. [46] proposed a discrimination function for finding the break locations for segmenting touching characters based on both pixel and profile projections. Lu [41] suggested a peak-to valley function to improve the above methods. The sum of differences between minimum value and the peaks on each side is calculated. The ratio of the sum to the minimum value itself (plus 1, presumably to avoid division by zero) is the discriminator used to select segmentation boundaries. This ratio exhibits a preference for low valley with high peaks on both sides.

Although most of the researchers have used these methods for character segmentation, and claimed reasonable segmentation accuracy, but these methods have some limitations. Hence, there is a need to develop new methods of segmentation which are robust to the degradation.

A number of algorithms have been proposed in the past for segmenting touching characters in Roman script [46-56]. Liang *et al.* [46] proposed a discrimination function for segmenting touching characters based on both pixel and profile projections. They proposed a dynamic recursive segmentation algorithm with an accuracy of 99.40-99.85% for segmenting touching characters. Wang and Jean [47] proposed a hybrid method for machine-printed character separation. They utilized a simple contour-tracing algorithm to initially separate the characters and applied a neural classifier to recognise the individual characters. In the case where two characters were not correctly separated by the simple dissection scheme, further segmentation schemes were invoked. Lee *et al.* [49] have segmented touching characters using projection profiles and topographic features extracted from the gray scale images. Then a nonlinear character segmentation path in each character segmentation region is found by using multi-stage graph search algorithm. Finally, recognition-based segmentation method is used to confirm the segmentation paths. Tsujimoto and Asada [50] constructed a decision tree for resolving ambiguity in segmenting touching characters. Authors used recognition result to identify multi character components. If a component had a similarity measure greater than a priori threshold, it was a single character component, otherwise it was a multi-character component. Casey and Nagy [48] proposed a recursive segmentation algorithm for segmenting touching characters. Hong [51] has utilized visual inter-word constraint available in a text image to split word images into pieces for segmenting degraded English language characters. In 1987, Kahan *et al.* [52] have proposed an objective function, the segmenting objective function, for finding breaking points within merged characters. Bose and Kuo [53] used a robust structural analysis technique based on line adjacency graph for segmenting touching characters. Prominent stroke and their directions were noted and arranged in ascending order. Two overlapping strokes with slopes of opposite sign is considered segmentation point. Schenkel and Jabri [54] have used combined segmentation and recognition technique (internal segmentation) for generating tentative character. A space displacement neural network is trained to generate character probabilities. The output of the neural network is then post-processed by a Hidden Markov Model that effectively searches through the recognition character candidates for optical character identification and boundaries. Zhao *et al.* [55] proposed a two-stage approach to segment unconstrained handwritten Chinese characters. In their approach, first a character string is coarsely

segmented based on the vertical projection and background skeleton, and the blocks of connected characters are identified; then in the fine segmentation stage the connected characters are separated with an accuracy of 81.6%. Lu [41] suggested a peak-to valley function to improve the above methods. Casey and Lecolinet [39] have discussed a few techniques for finding break locations in touching characters. Based on propagation and shrinking processes, an algorithm for partitioning touching characters has been discussed by Nakamura *et al*. [56].

2.1.2 Segmentation of touching numerals

Many algorithms [57-63] have also been proposed to segment touching handwritten numerals. Elnagar and Alhajj [57] proposed a thinning based algorithm for segmenting single touching handwritten digits, based on background and contour features in conjunction with a set of heuristics to determine the potential segmentation points with an accuracy of 96%. Donggang and Hong [58, 59] have developed a method to separate single touching handwritten numeral strings using structural features. In their approach, based on the structural points in the handwritten numerals strings, touching region of touching component is determined, and then based on the geometrical information of a special structural point, a candidate touching point is selected. Finally, they used morphological analysis and partial recognition results for the purpose. Chi *et al.* [60] proposed a contour curvature-based algorithm to segment single- and double- touching handwritten digit strings. Lu *et al.* [61] proposed a background thinning approach for the segmentation of connected handwritten digit strings. Pal *et al.* [62] have used water reservoir concept for segmenting unconstrained handwritten connected numerals. They have considered the location, size and touching position (top, middle or bottom) of the reservoir and then analyzed the reservoir boundary, touching position and topological features of touching pattern, determining the best segmentation point with an accuracy of 94.8%. Chen and Wang [63] used thinning based method to segment single- or multiple-touching handwritten numerals. They performed thinning of both foreground and background regions on the image of connected numerals strings. The end and fork points obtained by thinning are used for cutting points extraction.

2.1.3 Segmentation of touching characters in Indian scripts

Few algorithms have been investigated on segmenting touching characters in Indian scripts [64-69]. Bansal and Sinha [64] have segmented the conjuncts (one kind of touching patterns) in Devanagari script using the structural properties of the script. Conjuncts are normally found in Devanagari script pages. Conjuncts are basically a combination of a half character followed by a full character. Both the half and full characters always touch each other. So these kind of touching characters are due to structural properties of the script, but not due to other reasons which produces touching characters. As such, many problems, which are found during segmentation of touching characters produced due to other reasons (discussed in Chapter 3), are usually not found in segmenting these kinds of touching characters. In the technique proposed by the authors, first the left half character of the conjunct is segmented using the structural properties of the Devanagari script. They proposed that the half character is found in one-third to half portion of the full width of the conjunct. Starting from one-third column of the total width of the conjuncts, whenever one gets more number of pixels in a column then the previous column, that column marks the half character boundary. Now, by using the concept of collapsed horizontal projection, the left boundary of the second constituent character of the conjunct (which is a full character) is found. As claimed by the authors, the success rate of these algorithms for segmenting the conjuncts is 84%. The limitation of this work is that it will work only to segment the conjuncts and not the composite characters.

Garain and Chaudhuri [65] have used a technique based on fuzzy multifactorial analysis to segment touching characters in Devanagari and Bangla scripts. They proposed a predictive algorithm for effectively selecting possible cut columns for segmenting touching characters. The authors have claimed 98.92% accuracy for correctly segmenting touching characters. The limitation of their process of segmenting touching characters is the proposition that at touching position the width of touching blob is limited to few columns. This technique works fine if the width of touching blob is small but it does not work that well when the blob width at touching position is equal to or greater than the stroke width. Although, the simultaneous application of recognition and segmentation process will eliminate this problem to some extent, but performance may not improve drastically. We have tried to resolve this issue in

the Chapter 4. Earlier, Garain and Chaudhuri [66] have proposed three step rules to segment touching characters in printed Bangla script. First, to identify touching characters, aspect ratio analysis has been done along with recognition score. The component for which a normalized size invariant similarity measure is less than threshold value, it is suspected to be touching characters, provided that its bounding box aspect ratio is more than a predefined threshold. Second, to decide about the cut position degree of middle ness of touching position and thickness of the single black run is evaluated and dividing these two weights for the potential cut positions is evaluated. The cut position candidates are found at local maxima in the histogram of weights. Sometimes, it may produce over segmentation. Third, to eliminate the over segmentation each segmented characters is passed through recognition process. If the character is recognised then accepted, otherwise this segment is merged with the next segment and again the recognition process is applied, on the newly constructed segment and this process continues until recognition phase accepts it as a valid character. The authors have claimed an accuracy of 97% of segmenting touching characters, using this algorithm.

Lehal and Singh [67, 68] have given an algorithm to segment touching characters in upper zone of Gurmukhi script.

Chaudhuri *et al*. [69] have used the principle of water overflow, from a reservoir, to segment touching characters in Oriya script. The principle is that if we pour water on the top of a character, the positions where water accumulates are considered as reservoirs. Figure 2.2 shows the reservoir formed in a single character as well as in a pair of touching characters. A reservoir whose height is small and which lies in the upper part of the middle zone of a line is considered as a candidate reservoir for touching character segmentation. The cusp (lowermost point) of the reservoir of the candidate reservoir is considered as the separation point of touching characters. In Figure 2.2, a vertical line marks this position. Because of the round shape of the most of the Oriya characters, it was observed, that a reservoir is generally formed when two characters touch each other.

Figure 2.2: Water reservoirs (shown by dotted lines) in single and touching Oriya characters

Few works have also been reported for segmenting the horizontally overlapping lines in Indian scripts. Bansal [70] has discussed a two-pass algorithm based upon average line height to solve the problem of horizontally overlapping lines in Devanagari script. Harikumar *et al.* [71] have used the concept of average line height to segment the horizontally overlapping lines in Malayalam script. For segmenting unconstrained handwritten text lines of Bangla script, Pal and Datta [72] have divided the text into vertical strips and then taken the horizontal projections.

Pal *et al.* [73] have used various features of Indian scripts like existence of headline, number and position of peaks in horizontal projections, water reservoir *etc*. to separate various lines from multi script document. Pal and Chaudhuri [74] have used structural and statistical features for separating machine-printed text lines from handwritten text lines for both Bangla and Devanagari scripts. Dholakia *et al.* [75] have used slopes of connected components to find three zones in the printed Gujarati script. Pal and Chaudhuri [10, 76] have also discussed the concept of zoning and line segmentation. According to Lehal [77], the problem of segmenting multiple strips in Gurmukhi documents can be solved using average core strip height.

2.2 Feature Extraction

Feature extraction plays an important role in the successful recognition of machine-printed and handwritten characters [23, 78]. Feature extraction can be defined as the process of extracting distinctive information from the matrices of digitized characters. In OCR applications, it is important to extract those features that will enable the system to differentiate between all the character classes that exist. Many different types of features have been identified in the literature that may be used for character and numeral recognition.

Two main categories of features are Global (statistical) and Structural (topological) [78]. Global features are those that are extracted from every point of a character matrix. Initially, some global techniques were designed to recognise machine-printed characters [22]. Global features can be detected more easily and are not as sensitive to local noise or distortions as are topological features. However, in some cases small amount of noise may have an effect on the actual alignment of the character matrix, hence displacing features. This may have serious repercussions for the recognition of characters affected by these distortions [22, 78]. Global features themselves may be further divided into a number of categories. The first and most simple feature is the state of all the points in a character matrix. In a binary image there are only black or white pixels, the state therefore refers to whether a pixel is black or white. One strategy that has been mainly used for extraction of global features is based on the statistical distribution of points [23]. Six methods that have been employed in the literature, based on the distribution of points, are briefly outlined in next sub-section.

Trier *et al*. [79] summarised and compared some of the well-known feature extraction methods for off-line character recognition. Selection of a feature extraction method is probably the single most important factor in achieving high recognition performance in character recognition systems. They discussed feature extraction methods in terms of invariance properties, reconstructability and expected distortions and variability of the characters.

Besides the statistical and structural features, series expansion coefficients are also used as features of a character. The literature on these three categories is discussed below.

2.2.1 Statistical or global features

Statistical features are statistical measures of distribution of points on the bitmap, the contour curve, the profiles, or the HV-projections. Widely used methods are moments, zoning, *n*-tuples, projections, characteristic loci and crossings and distances.

(a) Moments: Moment invariants are features based on statistical moments of characters. A number of methods in this category utilize the moments of pixels in an image as features. They are traditionally used tools for character recognition [80-83]. Classical moment

invariants were introduced by Hu [84]. Hu's seven moments are well known to be invariant to position, size and orientation of the character. They are pure statistical measures of the pixel distribution around the center of gravity of the character and allow capturing the global character shape information. Higher order moments are difficult to apply. Several authors have proposed alternatives to Hu's moments (Radial and angular moments [85] and Zernike moments [86]). Another type of moments, namely, central moments are calculated by taking into account the distance of points from the centroid (centre of gravity) of the character [22, 23]. In this instance, central moments are preferred to raw moments as they produce higher recognition rates and are invariant to the translation of the image [23].

(b) Zoning: This method divides the character matrix into small windows or zones. The densities of points in each window are calculated and used as features to the chosen classifier. It was introduced in 1972 by Hussain *et al.* [87] and has been used by Bosker [12] for the commercial OCR system *Calera* and also by Messelodi and Modena[88].

(c) *n*-tuples: This method simply uses the occurrence of black or white pixels in a character image as features. The *n*-tuple method has been explored by Tarling and Rohwer [89]. Features extracted by the *n*-tuple scheme measure random properties of pixels.

(d) Projections: Projections are derived from histograms of horizontal and vertical projections of black pixels in some particular area of the character. Projections were introduced in 1956 in a hardware OCR system by Glauberman [90] and have been used by Heutte *et al.* [80] and Akiyama & Hagita [91].

(e) Characteristic loci: In this method, vertical and horizontal vectors are generated for each white background pixel in an image. Features are generated by counting the number of times a line segment is crossed in the vertical and horizontal direction.

(f) Crossings and distances: Lastly, researchers have obtained features by analyzing the number of times the character image is crossed by vectors in certain directions or angles, *i.e.*, 0^o, 45^o, 90^o *etc.* [22, 23].

2.2.2 Structural features

Structural features describe a pattern in terms of its topology and geometry by giving its global and local properties. Some of the main structural features include features like number and intersections between the character and straight lines, holes and concave arcs, number and position of end points and junctions [80]. These features are generally hand crafted by various authors for the kind of pattern to be classified. Chinese characters contain rich structural information, which remains unchanged over font and size variation. Since the basic elements of a Chinese character are strokes, the types and numbers of strokes and relationships among the strokes are essential structural features of a Chinese character. Lee and Chen [92] have represented each Chinese character by a set of short line segments, where each line segment is represented by its start and end point coordinates. The following three features are then extracted to represent a line segment: the center point coordinate, the slope and the relationships between the line segment and its neighboring line segments.

Amin [93] has used seven types of structural features such as number of subwords, number of peaks of each sub word, number of loops of each peak, number and position of complimentary characters, the height and width of each peak for recognition of printed Arabic text.

Lee and Gomes [94] have used the structural features for handwritten numeral recognition such as number of central, left and right cavities, location of each central cavity, the crossing sequences, the number of intersections with the principal and secondary axes and the pixel distribution.

Rocha and Pavlidis [95] have proposed a method for the recognition of multifont printed characters using the following structural features: convex arcs and strokes, singular points and their relationships. The singular point is one of the following places: a branch point, an ending point, a convex vertex and a sharp corner.

In a classic paper Kahan *et al.* [52] have developed a structural feature set for recognition of printed text of any font and size. The feature set includes the following information for a character: number of holes, location of holes, concavities in the skeletal structure, crossings of strokes, endpoints in the vertical direction and bounding box of the character.

Leedham and Pervouchine [96] have used global features like handwriting size, word spacing, line spacing, arrangement of words, margin patterns, baseline patterns, line quality, spelling and grammar *etc.* for recognition of handwritten text. The authors have used local features like character size, height-width ratios of characters, size and shape of loops, letter slant, letter design, letter spacing, writing pressure, speed variation, *t*-crossings and *i*-dots, hooking, punctuation marks, baseline patterns and angle, word slant, average stroke width, height of main body, ascenders, depth of descenders, loop features: area, slant *etc.* for the same purpose.

2.2.3 Series expansion coefficients

Need of decreasing the size of feature vectors while rendering the features immune to rotation and translation leads to discussion of the third global feature category: Transformations and Series expansions. The methods of detecting features by Transformations and series expansions have proven to be invariant to scaling, rotation and translation, while reducing dimensionality of the feature vector. Many researchers have used transformations and series expansions as features for the task of character recognition [21-23, 78]. The most common series expansion coefficient based statistical features rely on Fourier transform [97]. In Fourier transformation the characters are thresholded and their borders are extracted. The border of each character can be represented by its Fourier transform. Fourier coefficients with significant values are called Fourier descriptors and can be used as features for character recognition. The most straightforward application of the Fourier transform is to directly perform the two-dimensional version of the transform on the image and take the coefficients. The coefficients or more often combinations of coefficients, of these terms are used as features.

2.3 Classification

Classification is the second component of OCR engine as explained in Chapter 1. As already explained, classification is the component of the recognition system that attempts to detect the class that a particular character belongs to. Before a classifier may do this, it must be

shown a large number of training patterns, in a type of learning phase [23]. It has been noted that the key to high performance is through the ability to select and utilize the distinctive features of characters. No simple scheme is likely to achieve high recognition rate, hence more sophisticated systems have been developed. We have discussed in this section various types of classification methods that have been explored. List of various classification methods includes template matching, syntactic methods, statistical methods, artificial neural networks, kernel methods and hybrid classifiers [25, 32, 98].

2.3.1 Template matching

Template matching is one of the simplest and earliest approaches to patter recognition. In template matching, a template or a prototype of the pattern to be recognised is available. The pattern to be recognised is matched against the stored template while taking into account all allowable pose and scale changes [25]. Matching techniques can be grouped into three classes: direct matching, deformable templates and elastic matching and relaxation matching [32].

2.3.2 Syntactic or structural methods

In this kind of classifier an input pattern is classified in terms of its components (pattern primitives) and the relations among them. The classifier first identifies the primitives of a character and then parses strings of primitives according to a given set of syntax rules. Syntactic methods are mostly used for classifying handwritten text [25].

The most popular syntactic classification method is to represent characters as production rules whose left-hand side represents character labels and whose right-hand side represents string of primitives. The right-hand side of rules is compared to the string of primitives extracted from a word. A great deal of classification literature has been published in the area of decision tree and rule based learning techniques [99]. In decision trees, a character is represented syntactically as a tree whose internal nodes are primitives and leaves are character labels. Classifying a character, hence, corresponds to finding a path through the tree to a leaf. Rule-based systems are also used for classification of structural features [32, 38].

2.3.3 Statistical methods

Statistical classifiers consist of mapping fixed length vectors of features into a partitioned space [25, 32]. Classification here can be as simple as a distance classifier. Statistical classifiers are automatically trainable and, when reasonable assumptions are met, are relatively insensitive to noise. The *k*-NN rule is a non parametric recognition method. This method compares an unknown pattern to a set of patterns that have been previously labeled with class identities in the training stage. A pattern is identified to be of the class of the pattern to which it has the closest distance. Another common statistical method is to use Bayesian classification. A Bayesian classifier assigns a pattern to a class with the maximum a posteriori probability. Class prototypes are used in the training stage to estimate the class-conditional probability density function for a feature vector. Other then these, Quadratic Discriminant Function (QDF), Linear Discriminant Function (LDF), Euclidean distance, cross correlation, Mahanalobis distance, Regularized Discriminant Analysis (RDA) are other statistical classifiers used for classification. Hidden Markov Model (HMM) is a doubly stochastic process, with an underlying stochastic process that is not observable, but can be observed through another stochastic process that produces the sequence of observations. HMMs have been extensively applied to handwritten word recognition [100] and degraded text recognition [53, 54].

2.3.4 Artificial neural networks

A neural network is composed of several layers of interconnected elements called neurons. Each neuron calculates a weighted sum of its inputs producing an output signal that is transformed via the use of a linear or non-linear function. The main advantage of neural networks lies in the ability to be trained automatically from examples, good performance with noisy data, possible parallel implementation, and efficient tools for learning large databases. Neural network approach is non-algorithmic and is trainable. The most commonly used family of neural networks for pattern classification task is the feed-forward network, which includes multilayer perceptron and Radial-Basis Function (RBF) networks [101]. Some of the OCR systems which have used multi-layer feed-forward neural networks are

given by Cun *et al*. [102]. One problem with using neural networks in OCR is that it is difficult to analyze and fully comprehend the decision making process. Convolutional Neural Network, Vector Quantization (VQ) networks, auto-association networks, Learning Vector Quantization (LVQ) are other famous neural network methods used for classification purpose. The main weakness of the systems based on neural networks is their poor capability for generality. There is always a chance of under-training or over training the system. Besides this, a neural network does not provide structural description, which is vital from artificial intelligence viewpoint.

2.3.5 Kernel methods

Kernel methods, including Support Vector Machines [103], Kernel Principal Component Analysis (KPCA), Kernel Fisher Discriminant Analysis (KFDA) *etc*. are receiving increasing attention and have shown superior performance in pattern recognition. An SVM is basically a binary classifier with discriminant function being the weighted combination of kernel functions over all training samples. After learning by Quadratic Programming (QP), the samples of non-zero weights are called Support Vectors (SVs). For multi-class classification, binary SVMs are combined in either one against-others or one against-one (pair wise) scheme. Due to the high complexity of training and execution, SVM classifiers have been mostly applied to small category set problems. Promising results have been obtaining for handwritten digit recognition [104]. The use of SVM for recognising degraded text is also increasing.

2.3.6 Hybrid classifiers

All the above discussed classifiers have their own advantages and disadvantages. Combining multiple classifiers has been long pursued for improving the accuracy of single classifiers. Different classifiers tend to disagree on ambiguous patterns, so the combination of multiple classifiers can better identify and reject ambiguous patterns. Generally, combining complementary classifiers can improve the classification accuracy and the trade off between error rates and rejection rate. Parallel (horizontal) combination is more often adopted for high

accuracy, while sequential (cascaded, vertical) combination is mainly used for accelerating large category set classification. To improve recognition performance, especially for handwritten and cursive scripts such as Arabic and Indian language scripts, hybrid classifiers [105] have been used, which use diverse feature types and combinations of classifiers arranged in layers. It is based on the idea that classifiers with different methodologies or different features can complement each other. Hence if different classifiers cooperate with each other, group decisions may reduce errors drastically and achieve a higher performance. As a result, increasingly many researchers now use combinations of the above feature types and classification techniques. Baird [106] was one of the first researchers to propose a general technique for combining the strengths of structural shape analysis with statistical classification. The approach was to construct a function, called a feature identification mapping, from the representation generated by structural analysis to the one required for statistical classification.

2.4 Indian Script Recognition

As compared to English and Chinese languages, the research on OCR of Indian language scripts has not achieved that perfection. Few attempts have been carried out on the recognition of Indian character sets on Devanagari, Bangla, Tamil, Telugu, Oriya, Gurmukhi, Gujarati and Kannada. These attempts are briefly described in the following sub-sections.

2.4.1 Devanagari

Sinha [107-109] has started earlier work on Devanagari script recognition. He discussed a syntactic pattern analysis system and its application to Devanagari script recognition. Sinha and Mahabala [107] presented a syntactic pattern analysis system with an embedded picture language for the recognition of handwritten and machine-printed Devanagari characters. The system stores structural description for each symbol of the script in terms of primitives and their relationships. The recognition involves a search for unknown character primitives based on the stored description and context. To increase the accuracy of the system and reduce the computational costs, contextual information regarding the occurrences of certain primitives

and their combinations and restrictions are used. Sinha [108, 109] later suggested knowledge based contextual post-processing systems for Devanagari text recognition.

Sethi and Chatterjee [110] have also done some earlier work on Devanagari script. On the basis of presence or absence of some basic primitives, namely, horizontal line segment, vertical line segment, right slant and left slant, *D*-curve, *C*-curve *etc.* and their positions and interconnections, they presented a Devanagari hand-printed numeral recognition system based on binary decision tree classifier. They also used a similar technique for constrained hand-printed Devanagari character recognition [111]. Here, a set of very simple primitives is used, and all the Devanagari characters are looked upon as a concatenation of these primitives. A multi-stage decision process is used where most of the decisions are based on the presence / absence or positional relationship of the primitives.

Connel *et al*. [30] have developed an online Devanagari text recognition system. A Devanagari character recognition experiment with 20 different writers with each writer writing five samples of each character in a totally unconstrained way has been conducted by them. An accuracy of 86.5% with no rejects has been reported through the combination of multiple classifiers that focus on either local on-line properties, or global off-line properties. Bansal [70], has developed a knowledge based complete OCR for Devanagari script. Various relevant knowledge sources have been identified and integrated. A performance of 70% at character level has been achieved when the font is unknown. The performance improved to 80% when the font information is provided to the system. The use of word dictionary for correction further enhanced the performance to 90%.

Pal and Chaudhuri [10] have developed an OCR system for printed Devanagari script. Using zonal information and shape characteristics, the basic, modified and compound characters are separated for the convenience of classification. The modified and basic characters are recognised by a structural feature based tree classifier while the compound characters are recognised by a hybrid approach, which is a combination of a feature based tree classifier and a black run-length based dictionary look up. They have reported an accuracy of 96%.

Palit and Chaudhuri [112] have proposed a simple, feature-based algorithm for the computer recognition of printed Devanagari script. The algorithm uses a binary tree-structured classifier. The classifier uses three kinds of features. The first few levels are based

on features of a condensed run length method. At the lower levels, local features and moments are used. A recognition rate of 90% has been reported.

Biswas and Chatterjee [113] have presented a feature based approach to recognise Devanagari script documents using a combination of syntactic and deterministic approach. They have proposed two new features from characters that help in distinguishing between similar looking Hindi characters. One of the features employs the direction of the normal at the point of maximum curvature of a segment. The other feature is concerned with the direction of traversal of a segment. They have tested the system on six different fonts and have reported a recognition rate of 96%.

2.4.2 Bangla

After Devanagari, the next Indian language script on which work has been done is Bangla. One of the earliest attempts for Bengali character recognition has been made by Ray and Chatterjee [114]. They presented a nearest neighbor classifier employing features extracted by using a string connectivity criterion for Bangla character recognition. Exploiting the similarity among the major Indian scripts, Dutta [115] presented a generalized formal approach for generation and analysis of all Bangla and Devanagari characters. Dutta and Chaudhury [116] have developed an Isolated Optical Character Recognition (IOCR) system for Bangla alphabets and numerals using curvature features. Chaudhuri and Pal [76] have done extensive work for development of a complete OCR for Bangla script.

The first complete system capable of doing OCR from printed Bangla documents is due to Chaudhuri and Pal [76]. They have described an OCR system for documents of single Bangla font [9]. The characters are separated into simple and compound characters and are recognised separately. The simple character recognition is performed using a feature-based tree classifier, and the compound character recognition involves grouping. A recognition accuracy of 96% had been reported by the system. Later they improved the Bangla OCR system to increase the recognition accuracy of the system to 99.1% for single font clear documents [76]. From zonal information and shape characteristics, the basic, modified and compound characters are separated for the convenience of classification. The basic and modified characters are recognised by a structural-feature-based tree classifier. The

compound characters are recognised by a tree classifier followed by template-matching approach. A dictionary-based error-correction scheme has been used to increase the recognition accuracy. Chaudhuri and Pal [117] have also developed a skew detection technique for Indian language scripts such as Bangla and Devanagari, which exploits the inherent characteristics of the script to determine the skew angle. A word in these scripts appears as a single component. The upper envelop of a component is found by column wise scanning from an imaginary line above the component. Portions of upper envelop satisfying the properties of digital straight line are detected. They are clustered as belonging to single text lines. Estimates from individual clusters are combined to get the skew angle. The method works well for detecting skew angles in the range -45° to 45°.

Garain and Chaudhuri [118] proposed a method which combines the positive aspects of feature- and run number-based normalized template matching techniques, for the recognition of printed Bangla characters. For handwritten text recognition, Pal and Datta [119] proposed a water reservoir based scheme for the segmentation of unconstrained handwritten text into lines, words and characters. Neural network approach has also been used for the recognition of Bangla characters. Dutta and Chaudhury [116] reported a work on recognition of isolated Bangla alphanumeric handwritten characters using neural networks. The characters have been represented in terms of the primitives and structural constraints between the primitives imposed by the junctions present in the characters. The primitives have been characterized on the basis of the significant curvature events like curvature maxima, curvature minima and inflectional points observed in the characters. A two stage feed-forward neural net, trained by the well-known back-propagation algorithm, has been used for recognition. The structural constraints imposed by the junctions have been encoded in the topology of the network itself. Bhattacharya *et al*. [120] have also used neural network approach for the recognition of Bangla handwritten numerals. A topology adaptive self organizing neural network is first used to extract the skeletal shape from a numeral pattern. This skeletal shape is represented as a graph. Certain features like loops, junctions, *etc.* present in the graph are considered to classify a numeral into a smaller group. Finally, multilayer perceptron networks are used to classify different numerals uniquely.

Sural and Das [121] have developed a fuzzy OCR system for Bangla script. They tested their system on Bangla document images corrupted by simulated noise and have reported a recognition rate of 98% for Bangla documents.

2.4.3 Tamil

The work on recognition of Tamil characters started in 1978 by Siromony *et al*. [122]. They described a method for recognition of machine-printed Tamil characters using an encoded character string dictionary. The scheme employs string features extracted by row- and column- wise scanning of character matrix. Features in each row (column) are encoded suitably depending upon the complexity of the script to be recognised. Chandrasekaran *et al.* [123] used similar approach for constrained hand-printed Tamil character recognition. Chinnuswamy and Krishnamoorthy [124] presented an approach for hand-printed Tamil character recognition employing labeled graphs to describe structural composition of characters in terms of line-like primitives. Recognition is carried out by correlation matching of the labeled graph of the unknown character with that of the prototypes.

Recently a piece of work on on-line Tamil character recognition is reported by Aparna *et al.* [125]. They used shape based features including dot, line terminal, bumps and cusp. Stroke identification is done by comparing an unknown stroke with a database of strokes. Finite state automation has been used for character recognition with an accuracy of 71.32-91.5%.

2.4.4 Telugu

A two-stage recognition system for printed Telugu alphabets has been described by Rajasekaran and Deekshatulu [126]. In the first stage a directed curve tracing method is employed to recognise primitives and to extract basic character from the actual character pattern. In the second stage, the basic character is coded, and on the basis of the knowledge of the primitives and the basic character present in the input pattern, the classification is achieved by means of a decision tree. Lakshmi and Patvardhan [127] presented a Telugu OCR system for printed text of multiple sizes and multiple fonts. After pre-processing,

connected component approach is used for segmentation characters. Real valued direction features have been used for neural network based recognition system. The authors have claimed an accuracy of 98.6%. Negi *et al*. [128] presented a system for printed Telugu character recognition, using connected components and fringe distance based template matching for recognition. Fringe distances compare only the black pixels and their positions between the templates and the input images.

2.4.5 Oriya

In 1998, Mohanti [129] proposed a system to recognise alphabets of Oriya script, using kohonen neural network. The inputs pixels are fed to the neurons in the Kohonen layer where the neurons determine the output according to a weighted sum formula. The character is classified according to the largest output obtained from the neuron. Here the author experimented with only five Oriya characters and hence the reliability of the system is not established. In a system developed by Chaudhuri *et al*. [69] for the basic characters of Oriya script, the document image is first captured, preprocessed and segmentation modules are applied. These modules have been developed by combining conventional techniques with some newly proposed ones. Next, individual characters are recognised using a combination of stroke and run-number-based features, along with features obtained from their own designed concept of water over flow from a reservoir. Recently, Roy *et al.* [130] have presented a system for off-line unconstrained Oriya handwritten numerals. They used histograms of direction chain code of the contour points of the numerals as features and a neural network based classifier has been used with an accuracy of 94.81%.

2.4.6 Gurmukhi

Gurmukhi script is used primarily for writing Punjabi language. Punjabi Language is spoken by eighty four million native speakers and is the world's 14^{th} most widely spoken language. Lehal and Singh [11, 67, 68, 77, 131] developed a complete OCR system for printed Gurmukhi script where connected components are first segmented using thinning based approach. They started work with discussing useful pre-processing techniques [77]. Lehal

and Singh [67, 68] have discussed in detail the segmentation problems for Gurmukhi script. They have observed that horizontal projection method, which is the most commonly used method employed to extract the lines from the document, fails in many cases when applied to Gurmukhi text and results in over segmentation or under segmentation. The text image is broken into horizontal text strips using horizontal projection in each row. The gaps on the horizontal projection profiles are taken as separators between the text strips. Each text strip could represent: a) Core zone of one text line consisting of upper, middle zone and optionally lower zone (core strip), b) upper zone of a text line (upper strip), c) lower zone of a text line (lower strip), d) core zone of more than one text line (multi strip). Then using estimated average height of the core strip and its percentage they identify the type of each strip. For segmentation of strip into words, vertical projection is employed and a gap of two or more pixels in the histogram is taken to be the word delimiter. The word is then broken into sub-characters. First, the position of the headline in the word is found by looking for the most dominant row in the upper half of the word. Then, the connected component segmentation process proceeds in three stages and finally, each connected component represents either a single character or a part of character lying in one of upper, middle or lower zones.

In the recognition process, they have used two types of feature sets. In the primary feature set the number of junctions, number of loops and their positions are tested. The number of endpoints and their location, nature of profiles of different directions *etc.* are considered in the secondary feature set. A multi-stage classification scheme combined with binary tree and nearest neighbor classifier has been used for the purpose. The classification process is carried out in three stages. In the first stage, the characters are grouped into three sets depending on their zonal position, *i.e.*, upper zone, middle zone and lower zone. In the second stage, the characters in middle zone set are further distributed into smaller sub-sets by a binary decision tree using a set of robust and font independent features. In the third stage, the nearest neighbor classifier is used and the special features distinguishing the characters in each subset are used. One significant point of this scheme, in contrast to the conventional single-stage classifier where each character image is tested against all prototypes, is that a character image is tested against only certain subsets of classes at each stage. This enhances the computational efficiency. The system has an accuracy of about 97.34%. An OCR post-processor of Gurmukhi script is also developed. In last, Lehal and Singh [132] and Lehal *et*

al. [133] proposed a post-processor for Gurmukhi OCR where statistical information of Punjabi language syllable combinations, corpora look-up and certain heuristics based on Punjabi grammar rules have been considered.

There is also some literature dealing with segmentation of Gurmukhi Script [134, 135]. Lehal and Singh [134] have performed segmentation of Gurmukhi script by connected component analysis of a word assuming the headline not being a part of the word. Goyal *et al.* [135] have suggested a dissection based Gurmukhi character segmentation method which segments the characters in the different zones of a word by examining the vertical white space.

For segmenting touching characters, Lehal and Singh [10, 77] have used techniques to segment touching characters in Gurmukhi script in all the zones, *i.e.*, upper, middle and lower zone. For upper zone, they first applied segmenting objective function suggested by Kahan *et al.* [52]. Also they analyzed that if there is any eastward oriented stroke from a junction in the second half of the connected component (CC), along the x-axis, which is not touching the headline. If such a stroke exists, disconnect it from the main component. For middle zone, they used the technique used by Kahan *et al.* [52] on the unthinned characters. For lower zone characters touching with each other, they have used the same technique as used for upper zone.

In the presented work, we have proposed new algorithms, based upon the structural properties of Gurmukhi script, to segment touching characters in all three zones of degraded printed Gurmukhi script [136-138]. In the middle zone, various categories of touching characters have been identified. For segmenting touching characters falling in first category, we identified the position of the sidebar and put a cut column whenever the sidebar columns terminate. Similarly, algorithms have been developed to segment touching characters for other categories. This technique is very useful even for segmenting more than two touching characters simultaneously in a single word. Similar categories have been defined for upper and lower zones.

The problem of segmenting horizontally overlapping lines and associating the broken components of a line with their respective line has been studied and solved in printed Gurmukhi script and in seven other famous Indian scripts [139]. A survey has also been published about the problem of segmentation of touching characters in Indian scripts [140].

Different kinds of degradations in degraded printed Gurmukhi script have been identified along with the problems with their recognition, source of the degradations and some possible solutions have been discussed for solving the problems [141].

2.4.7 Gujarati

Antani and Agnihotri [142] described classification of a subset of printed Gujarati characters. For the classification, minimum Euclidean distance and *k*-NN classifier were used with regular and invariant moments. A Hamming distance classifier was also employed.

Sets of printed Gujarati characters and modifiers were chosen and subjected to classification by Yajnik and Mohan [143] using ANN architectures by considering linear activation functions in the output layer. The sample and test images for the Gujarati characters were obtained from the scanned images of printed Gujarati text and their features were extracted in terms of wavelet coefficients. Two Multi-Layer Perceptron (MLP) networks, one for the classification of alphabets which fall in the middle zone and the other one for classifying the modifiers which fall in the lower zone are designed. These networks achieve 94.46% and 96.32% accuracy for alphabets and modifiers, respectively, on the test set.

2.4.8 Kannada

Ashwin and Sastry [144] reported a font and size independent OCR system for printed Kannada documents. The proposed system first extracts words from the document image and then segments these into sub-character level pieces. The segmentation algorithm is motivated by the structures of the script. A set of zoning features is extracted after normalization of the characters for recognition. SVM has been used by employing a number of two class classifiers.

An on-line system for Kannada characters is described by Rao and Samuel [145]. The described system extracts wavelet features from the contour of the characters. The convolutional feed-forward multi-layer neural network is used as the classifier.

For recognition of machine-printed Kannada script, Kumar and Ramakrishnan [146] have studied ANN based classifiers like back propagation and Radial Basis Function (RBF) Networks, apart from the classical pattern classification technique of nearest neighbor. The ANN classifiers are trained in supervised mode using the transform features.

Sharma *et al.* [147] used quadratic classifier based scheme for the recognition of off-line handwritten numerals of Kannada.

2.5 Degraded Text Recognition

About degraded text recognition a very few work has been reported in the literature [51, 53, 54, 148]. For Indian languages almost nothing has been done on degraded text recognition. Hong [51] has done work in the degraded text recognition of English language. According to him in order to improve the performance of an OCR system on degraded images of text, post-processing techniques are critical. The objective of post-processing is to correct errors or to resolve ambiguities in OCR results by using contextual information. Depending on the extent of context used, there are different levels of post-processing. In current commercial OCR systems, word level post-processing methods, such as dictionary look up, have been applied successfully. However, many OCR errors cannot be corrected by word level post-processing. To overcome this limitation, passage level post-processing, in which global contextual information is utilized, is necessary. In most current studies on passage level post-processing, linguistic context is the major resource to be exploited. Relations at the image level must be consistent with the relations at the symbolic level if word image in the text have been interpreted correctly. Based on the fact that OCR results often violate this consistency, methods of visual consistency analysis are designed to detect and correct OCR errors.

A word-collocation-based relaxation algorithm and a probabilistic lattice-parsing algorithm are proposed. An interactive model for degraded text recognition is proposed that implement this strategy as shown in Figure 2.3. In this model, initial word recognition results are provided by an OCR system and they are treated as hypotheses to be tested further; by integrating visual and linguistic consistency analysis, a word hypothesis can be proposed,

modified, rejected, confirmed or selected; finally, a decision for each word image is determined.

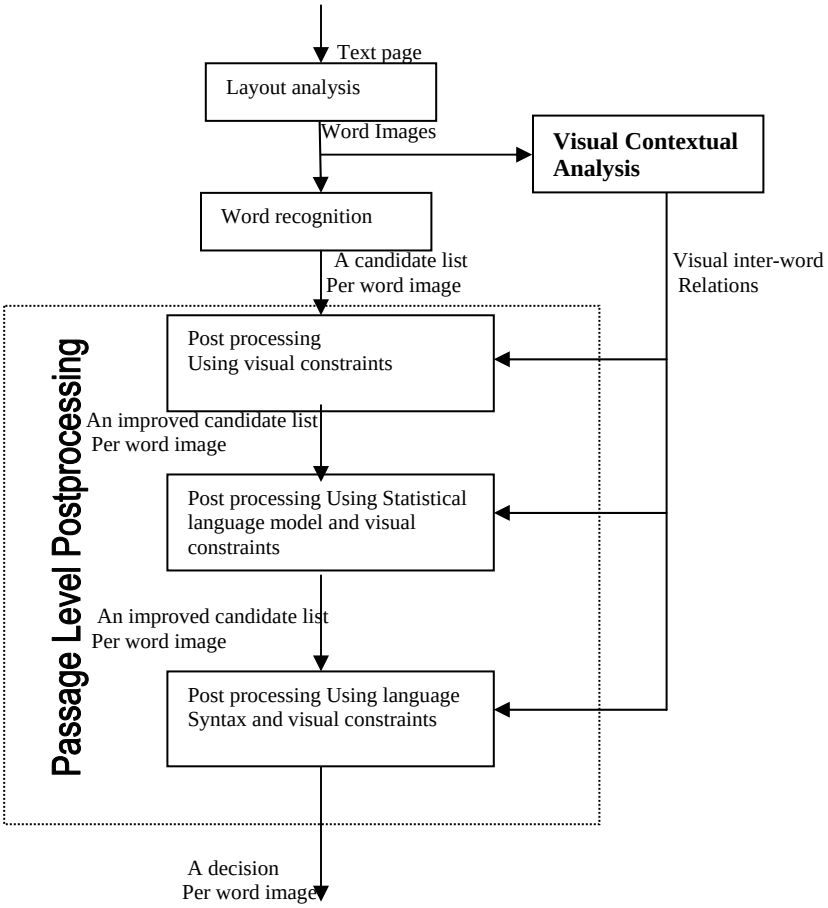


Figure 2.3: The Interactive model of text recognition system proposed by Hong [51]

Bose and Kuo [53] applied a Hidden Markov Model and level-building dynamic programming algorithm to the problems of robust machine recognition of connected and degraded characters forming words in a poor printed text. The recognition system consists of pre-processing, sub-character segmentation and feature extraction, followed by supervised learning or recognition. A structural analysis algorithm is used to segment a word into sub characters segments irrespective of the character boundaries, and to identify the primitive features in each segment such as strokes and arcs. The states of the HMM for each character are statistically represented by the sub-character segments, and the state characteristics are obtained by determining the state probability function, based on the training samples. In order to recognise an unknown word, sub-character segmentation and feature extraction are

performed and the transition probabilities between character models are used for the transition between characters in the string. A level-building dynamic programming algorithm combines segmentation and recognition of the word in one operation and chooses the best probable grouping of the characters for recognition of an unknown word. The computer experiments demonstrate the robustness and effectiveness of the new system for recognising words formed by degraded and connected characters.

Schenkel and Jabri [54] collected a large, real world database, containing degraded, old and faxed documents and present a comparison between two leading edge commercial software packages and human reading performance which shows quantitatively the huge performance gap between humans and machines, even on random character documents where no context can be used. This indicates room for possible improvements. They implemented an integrated segmentation and recognition algorithm using neural networks and HMM trained on the database and present results, which show the superior performance of the algorithm.

2.6 Degraded Text Recognition: A Challenge in OCR Research

Due to its complexity and variety of applications, OCR has become a very active field of research. The development of machines that can read printed and handwritten text, with a performance level similar to that of humans, has long been a goal with economic and scientific importance. Although a wide variety of techniques and many different approaches have been proposed over the last 40 years, the machine recognition of text still presents a challenge. Recently, some OCR systems with acceptable performance have been implemented for printed text in Roman script. As far as recognition of characters printed in Indian scripts is concerned, the OCR systems for only few scripts such as Bangla, Devanagari and Gurmukhi have been implemented with acceptable performance. These OCR systems have many challenging problems.

Designing and implementing an OCR for the recognition of degraded documents is one of the most challenging problems. Most of the work reported on Indian languages is on good quality documents. Degraded documents containing touching characters and heavily printed characters substantially decrease recognition accuracy. One of the most troubling and

difficult to model physical processes during printing, handling and scanning is noise. Noise usually affects the scanned document in a cumulative and quite unpredictable way. Some printer imperfection may cause blurry spots on the paper. Folding the paper may again create lines or shadows, when the paper is scanned. A document that has been faxed or copied several times is harder to read in comparison with the original document. Text gets thinner or thicker, salt and pepper noise appears, and contrast diminishes, glass imperfections, or dirt on the glass during scanning may create additional shadows and add foreign elements to the character image. Thermal noise and imperfections on the scanner photo cells may further alter the appearance of the document. It is very difficult to adequately model these processes. These are some of the difficulties in finding a good model for the degraded text recognition.

Chapter 3

Different Kinds of Degradations

Performance of an OCR system heavily depends on the print quality of the input document. OCR accuracy depends on document composition, printing, copying, and digitization. Even a small degradation which is negligible to human eye can be responsible for an enormous decline in the accuracy of an OCR system.

The recognition of machine-printed documents is possible with commercial OCR if the document images are sharp, noiseless and the characters are well written. As reported by Mori *et al.* [3], the activity of machine-printed character recognition is in progress since last sixty years. Currently, the recognition accuracy of the machine-printed characters is generally higher than 99.99%, which is sufficient for many real applications. However, there are several applications where the image quality is poor and standard OCR is not able to reach sufficient recognition capability.

Degradations in the scanned text image occur from many sources, which can be categorized into four areas as given below [149-151].

(a) Defects in the paper: Yellowing, wrinkles, coffee stains, speckles, typeface, point size, spacing and typesetting imperfections (pasted-up layouts, slipped type) *etc.* are the degradations that come in this category.

(b) Defects introduced during printing: Toner dropout, bleeding and scatter, baseline variations, ink-spread, strike-through and paper defects can be considered in this category.

(c) Defects introduced during digitization through scanning or camera: Skewness (geometric deformation), mis-thresholding, resolution reduction, blur, sensor sensitivity

noise, sampling, defocusing, low print contrast, non uniform illumination and non-rectilinear camera positioning are the degradations that can be put in this category.

(d) Defects introduced during copying through photocopiers and fax machines: Skew, streaking, shading and noise in electronics components can be considered in this category.

Degraded documents do not include all the ideal properties of a document. Li *et al.* [149] have discussed various defect models, their applications and methods for validating document defect models. Over the last few years, a lot of importance is being given to the problem of modeling document image defects so that a formal evaluation of the different OCR systems can be done. Most of the character recognition systems of today are found to be suitable for a specific type and quality of image. The methods and algorithms used in the development of these OCRs are often biased by the researcher's choice of the training and the test data sets. As a result, such systems give excellent performance for the data sets chosen by the researcher. In many cases, however, the recognition accuracy falls sharply when even a slightly degraded image is chosen. The fall in recognition accuracy is often high compared to the visual nature of the degradation, *i.e.*, the degradation as perceived by the human eye. It has, therefore, been felt necessary to model the defects quantitatively and experiment with extensive simulation to determine the nature of image defects that result in higher failure rate.

In this chapter, we have identified different kinds of degradations available in printed Gurmukhi script documents scanned from different sources. The sources of each kind of degradation, the problems associated with each kind of degraded text, comparison of each kind of degraded text in Gurmukhi script with the corresponding degraded text in Roman script, some possible solutions for identifying each kind of degraded text have been discussed in this chapter. We have also stated characteristics of Gurmukhi script, in particular, and other Indian scripts, in general.

3.1 Characteristics of Gurmukhi Script

Gurmukhi syllabary initially consisted of thirty two consonants, three vowel bearers, ten vowel modifiers (including *muktā* having no sign) and three auxiliary signs. Later on, six more consonants have been added to this script. These six consonants are multi-component characters that can be decomposed into isolated parts. Besides these, some characters modify the consonants once they are appended just below to them. These are called half characters or subjoined characters. The consonants, vowel bearers, additional consonants, vowel modifiers, auxiliary signs and half characters of Gurmukhi script (jointly called sub-symbols in this thesis) are given in Figure 3.1. In other words, a connected component after removing the headline is called a sub-symbol.

The Consonants

ਸ ਹ ਕ ਖ ਗ ਘ ਙ ਚ ਛ ਜ ਝ ਞ ਟ ਠ ਡ ਢ ਣ ਤ ਥ ਦ ਧ ਨ ਪ ਫ ਬ ਭ ਮ
ਯ ਰ ਲ ਵ ੜ

The Vowel Bearers

ੳ ਅ ੲ

The Additional Consonants (Multi Component Characters)

ਸ਼ ਜ਼ ਖ਼ ਫ਼ ਗ਼ .ਲ

The Vowel Modifiers

ਾ ਿ ੀ ੈ ੋ ੌ ੁ ੂ ੰ ੱ

Auxiliary Signs

ਁ ਂ ਃ

The Half Characters

ੑ ੍ ੵ

Figure 3.1: Characters and symbols of Gurmukhi script

Writing style in Gurmukhi is from left to right. The concept of capital and small characters is not there in Gurmukhi script. A line of Gurmukhi script can be partitioned into three horizontal zones, namely, upper, middle and lower zone. These three zones are described in Figure 3.2, with the help of one example word. The middle zone generally consists of the consonants. The upper and lower zones may contain parts of vowel modifiers auxiliary sings and half characters. In middle zone, most of the characters contain a horizontal line on the top, as shown in Figure 3.2. This line is called the headline. The characters in a word are connected through the headline along with vowel modifiers such as ◌ਿ, ◌ੀ, ◌ਾ *etc.* The headline helps in the recognition of script line positions and character segmentation. The segmentation problem for Gurmukhi script is entirely different from the scripts of other common languages such as English, Chinese and Urdu *etc.* In Roman script, windows enclosing each character composing a word do not usually share the same pixel values in vertical direction. But in Gurmukhi script, as shown in Figure 3.2, two or more characters of the same word may share the same pixel values in vertical direction. This adds to the complication of segmentation problem in Gurmukhi script. Because of these differences in the physical structure of Gurmukhi characters from those of Roman, Chinese, Japanese and Arabic scripts, the existing algorithms for character segmentation of these scripts do not work efficiently for Gurmukhi script.

Figure 3.2: Gurmukhi script word

With reference to Figure 3.2, line number 1 is called the start line, line number 2 defines start of the headline and line number 3 defines end of the headline. Also, line number 4 is called the base line and line number 5 is called the end line. Figure 3.2 also shows the contents of three zones, *i.e.,* upper, middle and lower zones. Region of the word in this figure from line number 1 to 2 encloses upper zone, from line number 3 to 4 contains middle zone and from line number 4 to 5 contains lower zone. Area from line number 2 to 3 defines width

of the headline. The upper and lower zones may remain empty for a word. These zones generally contain vowels, auxiliary signs and half characters.

3.2 Characteristics of other Indian Scripts

There are 23 official languages in India, namely, Assamese, Bengali, Bodo, Dogri, English, Gujarati, Hindi, Kannada, Kashmiri, Konkani, Maithili, Malayalam, Manipuri, Marathi, Nepali, Oriya, Punjabi, Sanskrit, Santhali, Sindhi, Tamil, Telugu and Urdu [4, 152]. There are 13 different scripts including Assamese, Bangla, Devanagari, Gujarati, Gurmukhi, Kannada, Kashmiri, Malayalam, Oriya, Roman, Tamil, Telugu and Urdu being used for writing these languages. The concept of upper/lower case characters is not present in Indian language scripts. Indian scripts can be divided into two major groups. First group having the concept of headline (a horizontal line at top of the characters), *e.g.*, Devanagari, Bangla and Gurmukhi script characters have headlines. Second group contains the scripts that do not have the concept of headline, *e.g.*, Gujarati, Kannada, Malayalam, Oriya, Tamil, Telugu *etc.* Figure 3.3 contains words from Gujarati script (non headline based script). In this figure, line number 1 is called the start line, line number 2 is the mean line, line number 3 is the base line and line number 4 is the end line. A text line of almost every Indian script can be partitioned into three horizontal zones, namely, upper, middle and lower zone except Urdu script.

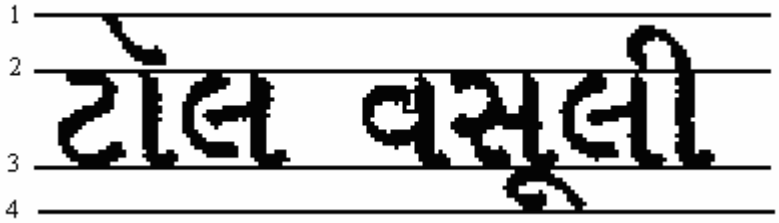


Figure 3.3: Gujarati script word

Figure 3.3 shows the contents of three zones, *i.e.*, upper zone (from line number 1 to 2), middle zone (from line number 2 to 3) and lower zone (from line number 3 to 4) in Gujarati script.

We have discussed the different kinds of degradations available in printed Gurmukhi script in next section.

3.3 Touching Characters

This is the most commonly found degradation in printed Gurmukhi script. In this category of degraded text, two neighboring characters touch each other. The important issue involved in recognition of touching characters is to segment them correctly, *i.e.*, identifying the position at which touching pair of characters must be segmented. Every OCR must perform well to the important task of separating them. The accuracy of an OCR depends heavily on the accuracy of segmentation process. The sources of documents containing touching characters are magazines with heavy printing, newspapers printed on low quality paper, very old books whose pages have turned yellow due to aging and photostatted documents copied on low quality machines. Figure 3.4 contains the Gurmukhi words containing touching characters.

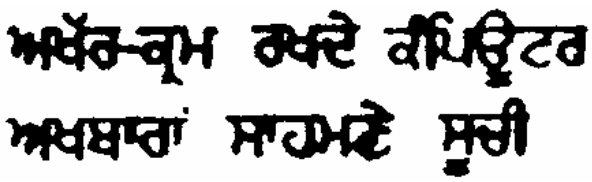


Figure 3.4: Touching characters in Gurmukhi script

Existence of touching characters in any document decreases the recognition accuracy of OCR drastically. On statistical analysis of touching characters, we have made the following observations [65].

- Touching characters are found in all three zones (*i.e.*, upper, middle and lower zones) of a degraded Gurmukhi script document.
- Touching characters touch each other mostly at the centre of the middle zone, less frequently at top of the middle zone and very less at the bottom of the middle zone.
- Most of the times, touching characters have larger aspect ratio than that of individual character.
- Generally, in a single word only two characters touch each other. The possibility of more than two touching characters is very less.
- Generally, the vertical thickness of the blob at touching position is small as compared

with the thickness of the stroke width. But in some cases, thickness may be equal or greater than the stroke width.

- Generally, the characters of Indian scripts contain sidebars at their right end, *e.g.*, in Gurmukhi script 12 consonants have side bars at their right end. The possibility of touching is very high at this position.

Segmentation of touching characters is most challenging task. There are two key issues involved in this problem. The first issue is to find the candidate of segmentation, *i.e.*, to find the segment of the complete word which may contain touching characters. Second issue is to find the break location within the candidate of segmentation, *i.e.*, to find the column which will correctly segment the two touching characters into isolated characters.

The problem of segmenting touching characters in Gurmukhi script is quite different from the Roman script in many aspects:

- In Gurmukhi script touching characters can be found in upper, middle and lower zone. Further touching characters can be divided into 5 categories.
  (a) Upper zone characters touching with each other (as shown in Figure 3.5(a)).
  (b) Upper zone characters touching with middle zone characters (as shown in Figure 3.5(b)).
  (c) Middle zone characters touching with each other (as shown in Figure 3.5(c)).
  (d) Middle zone characters touching with lower zone characters (as shown in Figure 3.5(d)).
  (e) The lower zone characters touching with each other (as shown in Figure 3.5(e)).

But in Roman script there is no concept of upper, middle and lower zones.

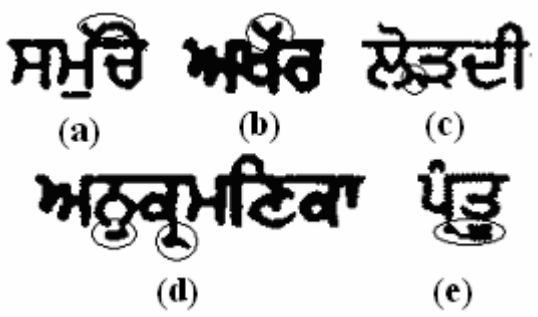


Figure 3.5: Touching characters in three zones (touching parts encircled): (a) upper zone characters touching each other, (b) upper zone characters touching with middle zone characters, (c) middle zone characters touching with each other, (d) middle zone characters touching with lower zone characters, (e) lower zone characters touching with each other

- Generally, the shape of the two touching characters in Gurmukhi script is very much different from any basic character, in contrary to Roman script where the combined shape of “r” and “n” makes “m”.
- In Gurmukhi script the tendency of touching of middle zone characters is more in middle of the characters and less at upper and bottom of the characters, but in Roman script it is more at upper and bottom of the characters and less in the middle of the characters.
- As shown in Figure 3.2, due to presence of the headline, characters are always connected with neighboring characters in Gurmukhi script, which does not happen in Roman script, since the concept of headline is not present in Roman script.
- Another peculiar problem found in Gurmukhi text, which is not there in Roman script, is that of characters touching across the neighboring lines. This introduces complexity not only in character segmentation but also in line segmentation. Figure 3.6 contains touching characters in neighboring lines.

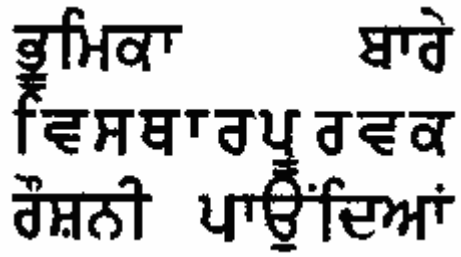


Figure 3.6: Document containing touching characters in neighboring lines

For segmenting touching characters in Gurmukhi script, the authors have proposed a solution discussed in detail in Chapter 4. The algorithms are proposed for segmenting touching characters in all three zones, *i.e.*, upper, middle and lower zones as well as touching characters in neighboring lines. These algorithms have shown reasonable improvement in segmenting touching characters.

3.4 Broken Characters

In this kind of degraded text a single character is broken into more than one component. It is also observed that fragmented characters cause more errors than touching characters or heavily printed characters. This may be a natural consequence of the fact that there are generally more white pixels, even in text areas of the page, than black pixels. Therefore, converting a black pixel to a white pixel loses more information than vice versa [18]. Figure 3.7a shows words of Gurmukhi script containing broken characters.

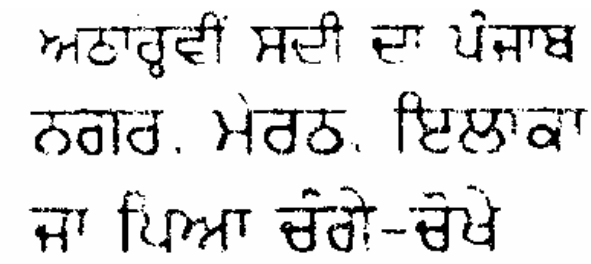


Figure 3.7a: Broken characters in Gurmukhi script

The main reasons of the occurrence of the fragmented or broken characters in the document are inadequate scanning threshold, tired printer or copier cartridges, worn ribbons, light printed magazines or documents, misadjusted impact printers, degraded historical documents, faxed documents, dot matrix text *etc.* Excessive fragmentation may destroy an entire phrase making it difficult to identify for human being. In the extreme cases, only a few pixels of a character remain, not even enough for a human to identify the character in isolation, as shown in Figure 3.7b.

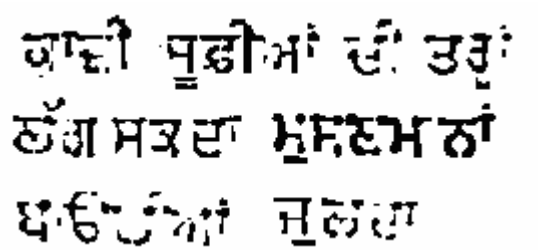


Figure 3.7b: Extremely broken characters in Gurmukhi script

Due to presence of broken characters the performance of any OCR may further decrease. Most of the work on recognition of headline based Indian scripts (Gurmukhi, Devanagari and Bangla) is based on the recognition of position of the headline. Due to broken characters, if headline is destroyed, as shown in Figure 3.7b, it will further make it difficult to identify headline, making the problem of character recognition more complicated.

On statistical analysis of broken characters we have made the following observations.

- One character may be broken either horizontally or vertically in more than one fragment. The percentage of horizontally fragmented characters is more than of vertically fragmented characters. This is due to that, generally the headline preserves it from breaking which causes less fragmented characters in vertical direction. Diagonally broken characters are also found in printed Gurmukhi script.
- If spacing between the characters is less, it becomes difficult to determine which fragment belongs to which character.
- Generally, each fragment of broken character will have aspect ratio less than of a single isolated character.
- Broken characters are generally found in middle zone, less in upper zone and very less in lower zone.
- The fragment of a character is generally not similar in shape of some other individual character.

The recognition of broken characters in Gurmukhi script is somewhat different from the Roman script.

- In Gurmukhi script, mainly broken characters are found in middle zone and less in upper and lower zone, but there is no concept of zones in Roman script.
- Generally, there is less information loss in case of Gurmukhi broken characters as the headline can be preserved even if document contains broken characters. Restoration of headline may cause characters to lose less information as compared to Roman script characters as the concept of headline is not present in Roman script.
- In Roman script, diagonally broken characters are also found along with horizontal and vertical broken characters. But the chances of founding the diagonally broken characters in Gurmukhi script are less.
- One broken component of character in Roman script may be same in shape to some other character but in Gurmukhi script it happens rarely.

Restoration of the text is the most important task in case of recognition of broken characters. Repeatedly application of dilation and erosion operations as pre-processing task can help to restore the image to some extent. Headline can be restored easily with the detection of its position. For segmentation of broken characters, an approach using simultaneous segmentation and recognition process can be useful. In this technique, after selecting each fragment, it can be processed for recognition as a character. If it is recognised, take the next segment. Otherwise, merge next segment with this segment and try to recognise both the segments as the fragments of a single character. Keep on adding the fragments until a character has been identified.

Not very much work has been reported to be done to recognise broken characters. Here also segmentation of broken characters is the first issue during recognition of broken characters. Lu [41] has discussed two methods segmenting broken characters, first by employing a merging procedure based on the estimated character width and intervals, and second is to combine character components based on the recognition results. Lu *et al.* [153] proposed an algorithm based on estimation procedure, a sequential merging procedure, as grouping procedure based on the estimated character width, and a decision procedure. Nakamura *et al.* [56] and Okamoto *et al.* [154] proposed a character segmentation algorithm for broken characters, based on propagation and shrinking in vertical and horizontal directions. Yanikoglu [155] has estimated the pitch of the text and guided the segmentation

of broken characters by the location of the pitch window, defined by the estimated pitch and the offset. Droettboom [156] used a technique based on graph combinatorics to rejoin the appropriate connected components. Kovacs [157] developed a real time application for reading low quality computer printouts containing alphanumeric characters. The author obtained strong error reduction by applying a two stage recognition strategy. In the first step a raw character recognition method is used based on the comparison of the unknown images with a reference set, using a k-nearest-neighbors algorithm. The second step takes into account the correlation among damaged characters, by increasing the reference set with the already recognised images of the same document. This process is iterated for reducing the error rate.

There is no reported work done in recognition of broken characters in any Indian script. Therefore, there is wide scope to work in this area.

3.5 Heavily Printed Characters

Sometimes even if the characters that are easily isolated, heavy print can distort their shapes, making them unidentifiable. It is very difficult to recognise a heavily printed character. The source of this kind of degradation is the same as that of first category, *i.e.*, touching characters. Figure 3.8 consists of some of heavily printed characters in Gurmukhi script.

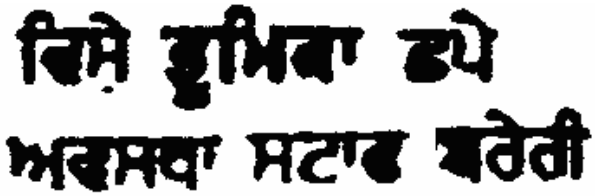


Figure 3.8: Heavily printed characters in Gurmukhi script

The following observations have been made on the statistical analysis of heavily printed Gurmukhi characters.

- The aspect ratio of heavily printed characters is almost same as that of the original character.
- It is very difficult to extract the features of a heavily printed character, as it is just like

a blob of pixels of the height and width of the original character, with no ascenders or descenders to help distinguish them.

- Generally, heavily printed characters are also touching with neighboring characters, *i.e.*, also falling in touching character category.
- Most of the characters, which are heavily printed have loop in their structure.
- Heavily printed characters can be found in middle zone as well as lower and upper zone also. Even in clean documents, characters in lower and upper zone are heavily printed.
- Most of the times, the shape of a heavily printed character may look like some other character.

The reasons of production of heavily printed characters are same as that of touching characters. As such, the problem of heavily printed characters is considered along with the problem of touching characters. Leading OCR of Roman script fails to recognise heavily printed characters [18]. Nothing specific has been done until now in Indian scripts to deal with the problem of heavily printed characters.

No special work has been reported for solving the problem of heavily printed characters. The best solution to recognise heavily printed characters is to bypass the recognition process until post-processing stage is encountered. Here, on the basis of dictionary look up if the word containing heavily printed characters is not a valid word, the dictionary look up post-processing process work will correct it automatically.

3.6 Faxed Documents

Fax machines are one of the major sources of text degradation. Faxed documents are treated as degraded documents as recognition of faxed documents create its own kind of problems. The process of sending documents by fax machine often results in distortions that are visible in the form of spurious point noise and ragged edges. Faxed documents are very light printed documents in general producing a large number of broken characters, few touching characters, and sometimes only few pixels remains of entire word. Faxed documents contain both salt and pepper noise. Sometimes it becomes difficult to recognise the faxed documents

text even for human being. An example from faxed document of Gurmukhi script has been show in Figure 3.9. The following observations have been made in the faxed document text.

- The width of the stroke is not constant over the document.
- Entire document contains varieties of the text, *i.e.*, broken characters, touching characters in all three zones, broken and merged characters *etc.*
- The quality the fax document also depends on the quality of fax machine.

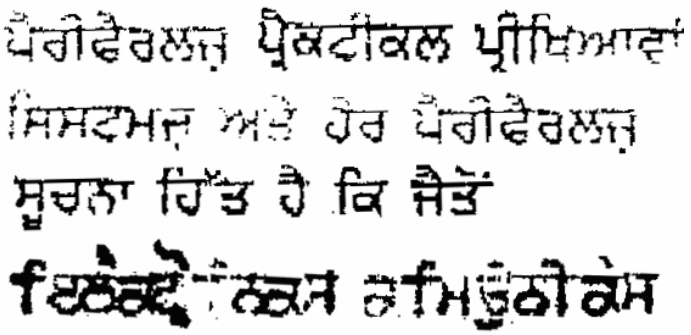


Figure 3.9: Words taken from faxed documents of Gurmukhi script

Some work has been reported to be done to enhance the faxed documents, so that they can be understood by a general OCR. Oguro *et al.* [158] have proposed three step solution for restoring faxed document by producing gray level images; determining the pixel value of pre-degraded images using the distribution of the neighboring pixels; and correcting the pixel value using the detected characteristics of sensor sensitivity. Randolph and Smith [159] developed a binary Directional Filter Bank (DFB) receiving a binary image and outputs a binary image comprised of directional components, and synthesis phase extracts the information from the output image. Hobby and Ho [160] have used bitmap clustering and averaging method for enhancing degraded documents especially fax documents. Natarajan *et al.* [161] have trained the system on degraded documents and then used adaptations to adjust the parameters of the trained model to improve the recognition accuracy.

3.7 Typewritten Documents

Typewritten documents are another kind of degraded documents. Typewriters are widely used in the government offices in India. An example of typewritten document in Gurmukhi script has been shown in Figure 3.10. Recognition of typewritten documents is itself a challenge as a typewritten document contains many problems.

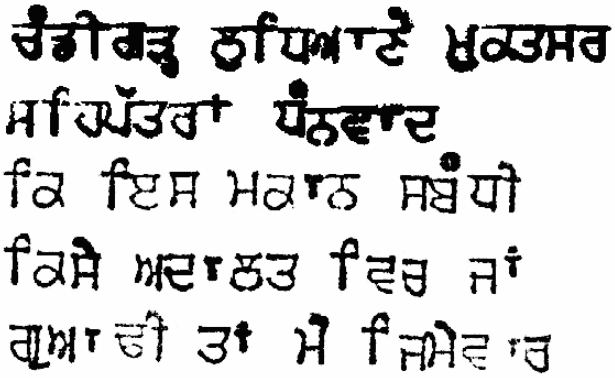


Figure 3.10: Words taken from typewritten documents of Gurmukhi script

On statistical analysis of the typewritten characters we have made the following observations.

- There are many touching characters in the middle zone of the typewritten text.
- Lower zone characters touch the upper zone characters almost every time. Hence segmentation of lower zone from middle zone is very difficult task.
- Unequal spacing between lines, words and even in the characters is observed.
- There is a significant change in the shape of upper zone characters.
- The headline of one complete word is usually broken in many parts and there are many ups and downs in the headline. Also, as shown in Figure 3.10, most of the times characters in a single word are separate from each other and their headline do not touch. It makes difficult to recognise headline and baseline. As most of the algorithms are designed on the basis of headline and baseline, it will lead to decrease in recognition accuracy.

- The darkness of the character depends on the force applied for striking the key of typewriter. Due to hard pressing generally characters are heavily printed and the shape of characters is distorted.
- The typewriters can be of fixed width grid or variable width grid. Any typewriter with fixed width grid produces characters occupying the same amount of horizontal space, whatever is the actual shape of the character.

Cannon *et al.* [162, 163] have suggested a method for automatically improving the quality of degraded images in a typewritten archive. Rodríguez *et al.* [164] have developed a new cost function to segment degraded typewritten digits.

3.8 Documents containing Backside Text Visible

Sometimes back side text is visible from the front side of text document. This kind of degradation is also called show thought or bleeds through problem. The former is due to the use of thin paper while the latter is caused by the seeping of the ink through the document page. It happens when the quality of paper is poor or printing is very dark. An example from Gurmukhi script document containing backside text visible has been show in Figure 3.11.

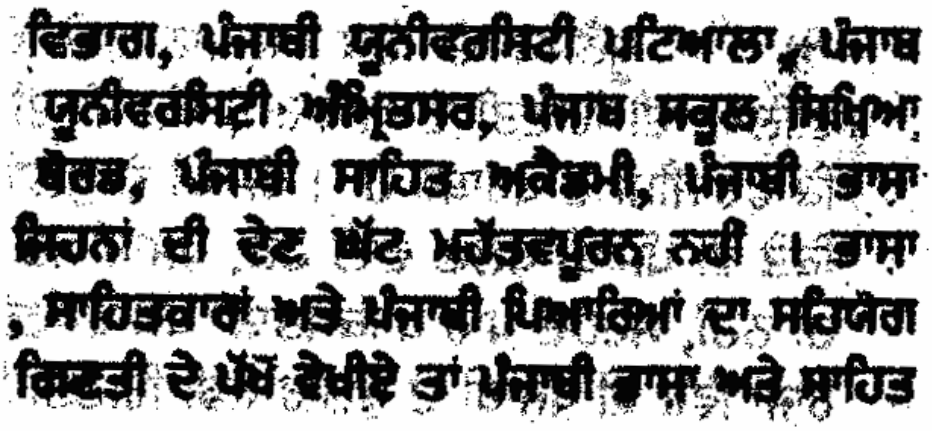


Figure 3.11: Gurmukhi script document containing backside text visible

On statistical analysis of the documents containing backside text visible characters, we have made the following observations:

- A lot of noise pixels are produced during binarization process of these documents.
- Line, word and characters segmentation methods fail at most of the places.

No specific work has been reported to be done to solve this problem. More suitable algorithms can be developed to remove the noise produced in such documents.

3.9 Discussion

Degraded text is a common problem in printed documents. Making an OCR which deals with degraded text is the need of the time. In this chapter, we have discussed many kinds of degradation available in printed Gurmukhi script. Mainly, in a degraded document, touching characters, broken characters and heavily printed characters are found. Faxed documents, typewritten documents and documents containing backside text visible are also considered as degraded documents. There are several kinds of other typographic degradations like decorative characters, underline characters, reverse video *etc.*

In this thesis, we have considered Gurmukhi script documents containing touching characters and heavily printed characters only. The reason of working on these two kinds of degradation is that touching characters and heavily printed characters are found in all kinds of degradations like faxed documents, typewritten documents and documents in which backside printing is visible. Hence, working on these two kinds of degradation is very useful. The algorithms proposed in this thesis may also work efficiently for solving the recognition problems associated with other kind of degradations.

Similar kinds of degradations are also available in other printed Indian scripts. Hence, a lot of research can be carried out to handle these kinds of the degradations in other Indian scripts as well.

Chapter 4

Segmentation

Segmentation is an important step towards designing an OCR system. In this chapter, we have proposed some new and efficient algorithms for line and character segmentation in degraded printed Gurmukhi script documents. A brief overview of actual system design followed and implemented by us in this thesis for designing an OCR system for degraded printed Gurmukhi script is presented in the next section.

4.1 System Overview

We have followed the system model as shown in Figure 4.1, for accomplishing the task of recognising degraded text of Gurmukhi script. For a degraded machine-printed Gurmukhi script document, we have used standard procedures for digitization that give digitized text image. After that, pre-processing activities such as noise removal, skew correction and thinning are performed using standard algorithms given in literature. We have worked on both thinned and unthinned data.

It can be noted from Figure 4.1 that there are four main processing stages: pre-processing, segmentation, feature extraction and recognition. In this chapter, we have discussed in detail the segmentation stage. Feature extraction and recognition stages will be discussed in the forthcoming chapters. We start with a brief overview of the pre-processing activity.

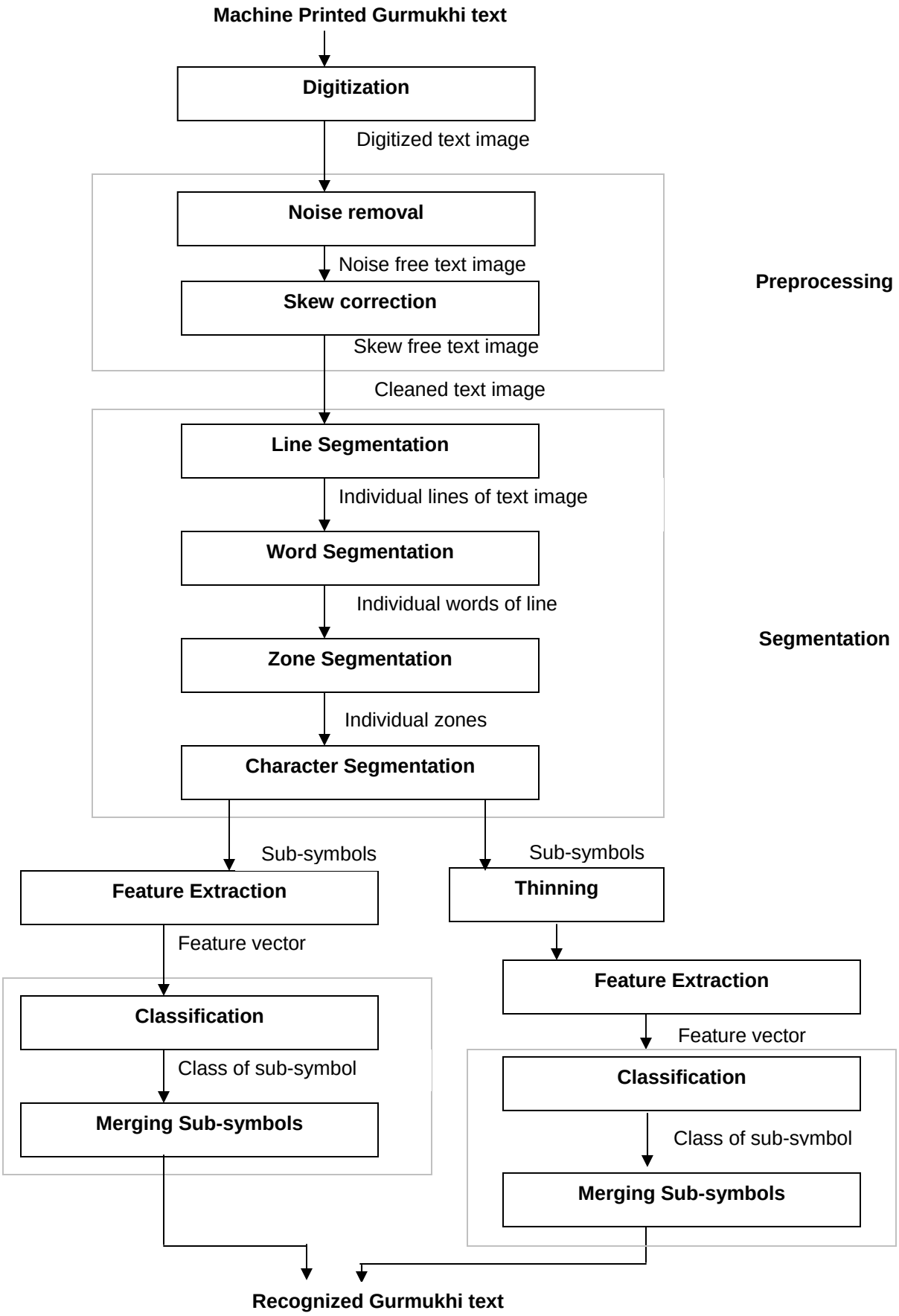


Figure 4.1: System model for recognizing degraded printed Gurmukhi text

4.2 Pre-processing

In this thesis, we have used the existing techniques of pre-processing. Noise removal algorithms have been applied on the input binary document in order to minimize the effect of spurious noise in the subsequent processing stages. In the present study, both salt and pepper noise have been removed using standard algorithm given by Iliescu *et al.* [165]. It has been noted that there is always some extent of skewness present in the document image that may lead to the poor performance of an OCR system. Therefore, the skewness has been removed with the help of standard skew detection and removal algorithm given by Chaudhuri and Pal [117]. The new algorithms proposed for segmentation in the present study do not perform that accurately in case the image is skewed, because the algorithms are based on the exact position of the headline and baseline of each line in the input document. If the image is skewed, it becomes very difficult to extract the exact position of the headline and the baseline and the algorithms do not work. We have, in some cases, performed thinning on input documents using standard algorithm by Zhang and Suen [166]. It is also worth to mention here that we have performed most of the experiments on unthinned data. Thinning has been performed on segmented characters before extracting the features of the characters.

4.3 Segmentation Process

Segmentation is one of the most important phases in character recognition process. Segmentation is the process of segmenting the whole document image into recognizable units for feature extractor and classifier. Text area from the document is extracted and the segmentation step is followed by segmenting the text region into individual lines. Further, each line is segmented into individual words, and finally, each word is segmented into individual characters.

Character segmentation is generally ignored by the research community, yet broken characters and touching characters are responsible for the majority of errors in automatic reading of both machine-printed and handwritten texts. Character segmentation is fundamental to character recognition approaches which rely on isolated characters. It is a critical step because most recognition errors are due to the incorrect segmentation of the

characters, especially when input document contains degraded text, *i.e.*, touching characters. The well known tests for commercial printed text OCR systems by the University of Neveda, Las Vegas, consistently ascribe a high proportion of errors to segmentation [39]. Bosker [12] has also inferred that there is a drop in recognition rate due to incorrect segmentation when touching characters are present in the document.

Character segmentation plays a very important role in a text recognition system. The simple technique of using inter-character gap for segmentation is generally useful for good quality printed documents, but this technique fails to give satisfactory results if the input text contains touching characters. We have proposed algorithms to segment touching characters and also to segment overlapping lines (produced due to touching characters in neighboring lines) in degraded printed Gurmukhi script. Various categories of touching characters in different zones, along with their solutions have been discussed in the forthcoming sections.

An input document may contain several kinds of information like photographs, figures, multiple articles (possibly in multiple columns) *etc.* Three major components of a complete text reading system are document analysis, document understanding and character segmentation [45]. The document analysis component extracts lines of text from a page for recognition. It also finds the constituents of a document such as photographs, graphics and text lines. The document understanding component extracts logical relationships between the document constituents. The character segmentation component extracts characters from a text line and recognises them. As such, for an OCR system, segmentation process involves following steps:

1. Detection of text regions in the input document.
2. Segmentation of text region into individual lines.
3. Segmentation of text line into individual words and zones.
4. Segmentation of word into individual characters.

These steps have been shown in Figure 4.2. Figure 4.2(a) contains an input document of Gurmukhi script. Figure 4.2(b) contains the text regions extracted from the input document. A text region has been segmented into lines in Figure 4.2(c). Figure 4.2(d) contains some of

the words extracted from a line. Also, in Figure 4.2(e), words have been segmented into characters.

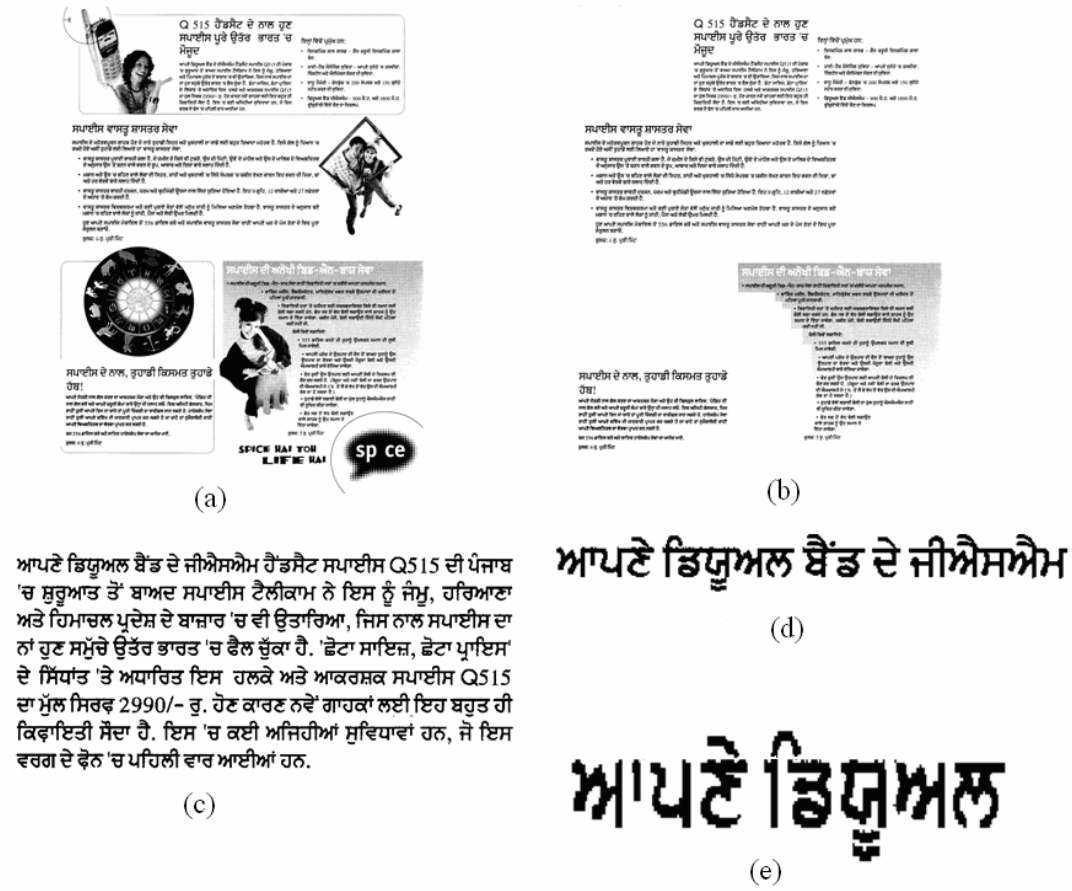


Figure 4.2: Segmentation process: (a) input document, (b) text regions extracted from input document, (c) segmented lines, (d) segmented words, (e) segmented characters

First step of segmentation process is detection of text regions in the input document. We have hypothesized in the present work that the text and graphics have already been separated and documents contain only single column text regions for experimental purposes.

4.4 Line Segmentation

Second step of segmentation process is segmenting the text region into lines, also called as line segmentation. Generally, each text line is separated from the previous and following

lines by white spaces. Therefore, the horizontal projection of a document image is the most commonly used technique to extract the lines from the document [10, 76, 133, 135]. If the lines are well separated and not tilted, the horizontal projection will have well separated peaks and valleys [77]. These valleys can be detected easily and used to determine the locations of the line boundaries.

In degraded printed Gurmukhi script, applying the simple concept of horizontal projection to segment the whole document into individual lines does not work well. Over segmentation occurs when the white space breaks a text line into two or more horizontal text strips as shown in Figure 4.3 (problem areas have been encircled). Sometimes, lower zone characters of one line touch the upper zone characters of next line, thus producing horizontally overlapping lines, called under segmentation, as shown in Figure 4.4 (problem areas have been encircled). The problem of horizontally overlapping lines is a common problem in newspapers and magazines of printed Gurmukhi script.

Figure 4.3: Horizontal projection of Gurmukhi script document resulting over segmentation

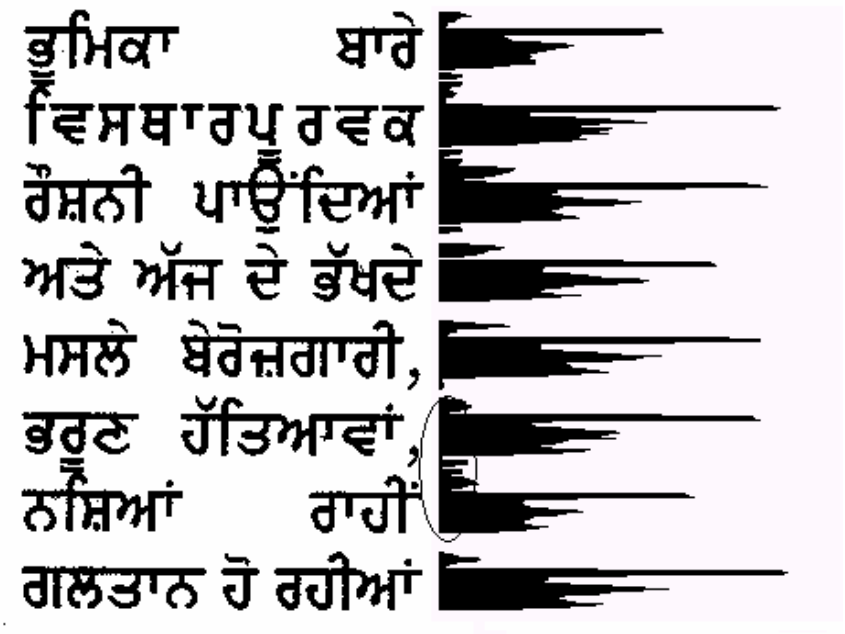


Figure 4.4: Horizontal projection of Gurmukhi script document resulting under segmentation.

Thus line segmentation becomes an important step in the segmentation process. There are many limitations with the previously used techniques for segmenting the lines. Dholkia *et al.* [75] have assumed that the lines have been segmented properly before finding three zones of a line from a Gujarati script document. This is not possible when the document contains horizontally overlapping lines. It is worth to mention here that no author has discussed in detail the problem of over segmentation of lines. The fragments of a line produced due to over segmentation make the problem of segmentation of lines very hard. The methods discussed in literature for segmenting overlapping lines fail when there is over segmentation [70, 71]. Furthermore, these methods do not work that accurately when there are more than two overlapping lines [70, 71]. Lehal and Singh [67, 68, 77] have discussed the methods of line, word and character segmentation. However, they have not discussed the methods for segmentation of overlapping lines. This work is the first attempt for discussing the problem of over segmentation in Gurmukhi script.

Before discussing the problem of multiple horizontally overlapping lines in printed Gurmukhi script and proposing its solution, we hereby give some definitions and notations used in various algorithms proposed in this chapter.

(a) Horizontal Projection: For a given binary image of size $L \times M$, where L is the height and M is the width of the image, the horizontal projection is defined by Bansal [64] as:

$HP(i)$, $i = 1, 2, 3, \ldots, L$.

where $HP(i)$ is the total number of black pixels in i^{th} horizontal row.

(b) Vertical Projection: For a given binary image of size $L \times M$, where L is the height and M is the width of the image, the vertical projection is defined by Bansal [64] as:

$VP(j)$, $j = 1, 2, 3, \ldots, M$.

where $VP(j)$ is the total number of black pixels in j^{th} vertical column.

(c) Continuous Vertical Projection: For a given binary image of size $L \times M$, where L is the height and M is the width of the image, the continuous vertical projection has been defined as:

$CVP(k)$, $k = 1, 2, 3, \ldots, M$

where $CVP(k)$ counts the first run of consecutive black pixels in k^{th} vertical column.

(d) Strip: A strip can be defined as a collection of consecutive run of horizontal rows, each containing at least one pixel.

The problem of line segmentation further intensifies in degraded printed Gurmukhi script as the horizontal projection of the document divides the whole document into following types of strips:

Type 1: strip containing only upper zone characters (strip number 1 in Figure 4.5a).

Type 2: strip containing only middle zone characters having upper zone and/or lower zone but this upper and/or lower zone has been segmented as the part of some other strip (strips number 2 and 6 in Figure 4.5a).

Type 3: strip containing upper zone characters touching with middle zone characters having no lower zone characters, *i.e.*, one line (strip number 3 in Figure 4.5a).

Type 4: strip containing upper zone characters touching with middle zone characters having lower zone characters, but the lower zone has been segmented into different strip (strip number 9 in Figure 4.5a).

Type 5: strip containing upper, middle and lower zone characters, *i.e.*, complete one line (strip number 4 in Figure 4.5a).

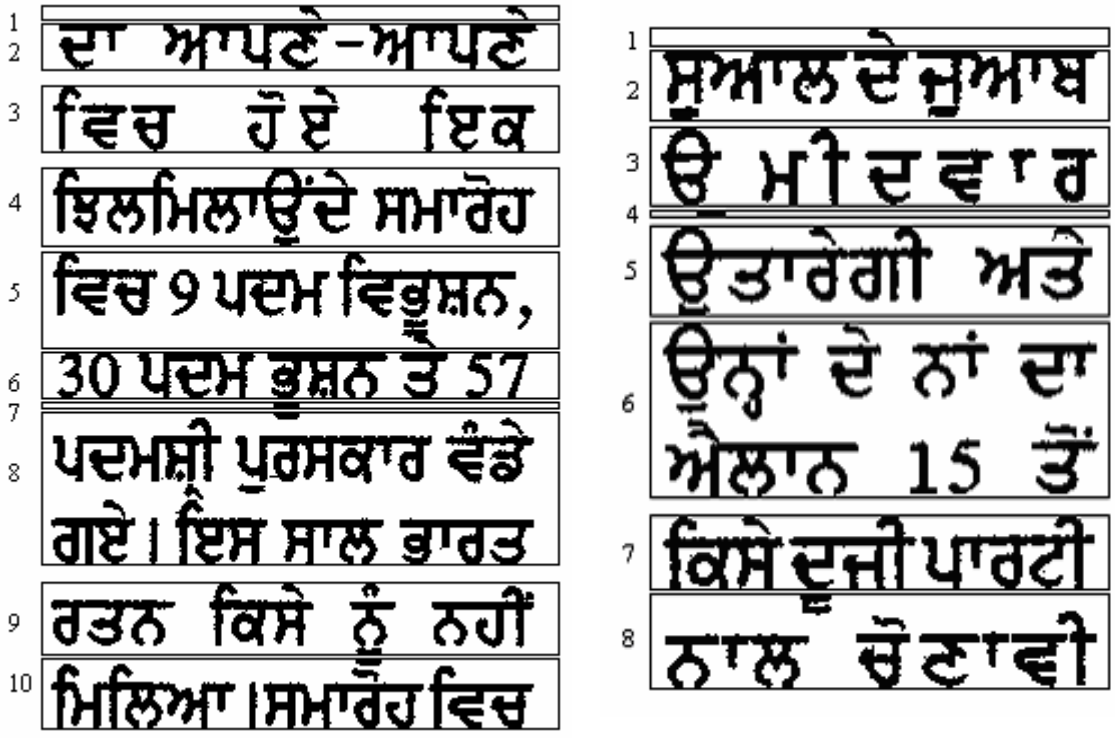
Type 6: strip containing upper, middle and lower zone characters of one line and upper zone of next line (strip number 5 in Figure 4.5a).

Type 7: strip containing only lower zone characters (strip number 7 in Figure 4.5a).

Type 8: strip containing lower zone characters touching with upper zone of next line (strip number 8 in Figure 4.5a).

Type 9: strip containing middle zone and lower zone characters whose upper zone have been segmented with some different strip (strip number 2 in Figure 4.5b).

Type 10: strip containing two or more horizontally overlapping lines (strip number 8 in Figure 4.5a).



(a) **(b)**

Figure 4.5: Various types of strips in degraded printed Gurmukhi script

It may be noted that there are ten strips in Figure 4.5a, but actual number of lines are nine. Strips numbered 3 and 4 constitute a complete line and need no segmentation. Strip 8 contains overlapping lines, which requires proper segmentation. Strips numbered 1 and 7 contain components from upper and lower zone and require the decision that these are part of which strip in order to make a complete line. These are named as components of over segmented text lines. Similarly, strips numbered 2 and 6, which contain only middle zone, need to include their upper and lower zones. Strips numbered 5 and 10 contain touching lower and/or upper zone of some neighboring lines. As a result of this, it is necessary to find the exact boundaries of these lines.

These different types of strips make it very difficult to find the category of a given strip. Besides this, in case of a strip containing horizontally overlapping lines, it is difficult to estimate the exact position of pixel row, which segments one line from the next line. Statistical analysis of degraded printed documents of Gurmukhi script reveals the information as shown in Table 4.1 about percentage occurrence of various strips.

Table 4.1: Percentage of occurrence of various types of strips in printed newspaper of Gurmukhi script

Strip type	Percentage of occurrence
Type 1	0.66
Type 2	0.78
Type 3	10.04
Type 4	20.78
Type 5	12.98
Type 6	2.04
Type 7	19.34
Type 8	14.84
Type 9	0.62
Type 10	17.92

These results have been obtained by analyzing one hundred thirty four documents, scanned from good quality printed newspaper articles. Two of the documents are shown in

Figure 4.5. In the next sub-section, we illustrate the algorithms for segmenting overlapping lines of uniform font size.

4.4.1 Segmentation of overlapping lines of uniform size

Most of the printed area of a document contains uniform sized text. We propose algorithm 4.1 (**Segment _Lines)** for segmenting overlapping lines of uniform size and joining together the components of the over segmented text lines of Gurmukhi script. This algorithm segments the whole document into individual lines. The basic idea behind this algorithm is that, in general, the difference between base line of one line and starting line of headline of next line is the same as the height of middle zone and each line has height of lower and upper zone approximately equal to half of the height of middle zone. The input of this algorithm is digitized binary matrix of a single column document of Gurmukhi script and its output is a document with proper line boundaries. Algorithm 4.1a contains the outline of the algorithm, while Algorithm 4.1b contains the detailed algorithm for line segmentation.

Algorithm 4.1a: Segment _Lines (document matrix in binary form)

begin

get binary matrix of the input document;
find all the strips and compute the height of each strip;
find the positions of all headlines;
compute average line height from the positions of headlines;
for (all strips)
if (height of a strip is less than 30% of average line height)
strip belongs to type 1, do nothing and loop for the next strip;
end-if
if (height of strip is greater than 50% of average line height)
strip belongs to type 2, 3, 4, 5, 6, 8, 9 or 10;
compute baseline position;
compute height of the middle zone from headline and baseline positions;

```
            compute half of the height of the middle zone and add it to baseline row for
            finding actual line boundary; //type 2, 3, 4, 5, 6, 8 and 9 solved
            if (strip boundary is greater than actual line boundary)
               strip belongs to type 10
               decrease height of the strip by one complete line;
               loop for the same strip;
            end-if
            if (strip boundary is less than the actual line boundary)
               strip belongs to type 7
               repeat
                     consider next strip;
               until (strip boundary is less than actual line boundary);
            end-if
         end-if
      end-for
end-algorithm
```

Algorithm 4.1b (Segment _Lines): segmentation of uniform sized text lines

BEGIN

Step 1: Using the horizontal projections, different strips denoted by $S_1, S_2, S_3, ..., S_m$ in input binary document are identified. For that, whenever $HP(i) = 0$ and $HP(i+1) \neq 0$ for $i = 1, 2, 3, \ldots, L$, it is marked as start of the strip line and called first row of strip denoted by $FR(S_p)$. Whenever, $HP(i) = 0$ and $HP(i-1) \neq 0$, for $i = 1, 2, 3, \ldots, L$, it is marked as the end of the strip line or last row of strip and denoted by $LR(S_p)$. Height of the strip is calculated as $H(S_p) = LR(S_p) - FR(S_p) + 1$, for $p = 1, 2, 3, \ldots, k$. Strips identified in a document are shown in Figure 4.5a.

Step 2: In order to identify the position of headlines, find

$MAXPIX = \max \{HP(i)\}$, $i = 1, 2, 3, \ldots, L$

The headlines are considered as those lines, whose $HP(i) \geq 70\%$ of $MAXPIX$ (the threshold limit of 70% is arrived at after detailed and careful experimentation). Whenever $HP(i) > 70\%$ of MAXPIX and $HP(i-1) < 70\%$ of MAXPIX, denote

it as starting position of the headlines as $SH_1, SH_2, SH_3, \ldots, SH_n$. Whenever $HP(i) > 70\%$ of MAXPIX and $HP(i+1) < 70\%$ of MAXPIX, denote it as the ending position of the headlines as $EH_1, EH_2, EH_3, \ldots, EH_n$. Also denote the lines to be identified as $L_1, L_2, L_3, \ldots, L_n$ (number of headlines are same as number of lines).

Step 3: Define

$$AVG_LINE_HEIGHT = \frac{1}{n-1}\sum_{i=2}^{n}(EH_i - EH_{i-1})$$

Step 4: Set $LINE_NO = 1$ and first row of line $LINE_NO$ as first row of first strip, *i.e.*, $FR(L_{LINE_NO}) = FR(S_1)$.

Step 5: For $i = 1$ to k, perform the following operations:

{

Step 5.1: if $H(S_i) < 30\%$ of AVG_LINE_HEIGHT, S_i is of type 1 (contains only upper zone), goto step 5

//ignore current strip and go for next strip

Step 5.2: if $H(S_i) > 50\%$ of AVG_LINE_HEIGHT, S_i will be of type 2, 3, 4, 5, 6, 8, 9 or 10 and will contain at least one headline and one baseline.

Step 5.3: identify the position of baseline by noting the $CVP(k)$, for $k = EH_{LINE_NO}$ to $LR(S_i)$. The position where $CVP(k)$ ends, mark it as α, every time. The row in which maximum α are found is considered to be the baseline. Mark it as $BASE_{LINE_NO}$. Also set height of the middle zone as $HGT_MID = BASE_{LINE_NO} - EH_{LINE_NO}$.

Step 5.4: set last row of line $LINE_NO$ as $LR(L_{LINE_NO}) = BASE_{LINE_NO} + (HGT_MID) / 2$.

//strip type number 2, 3, 4, 5, 6, 8 and 9 solved here.

Step 5.5: if $(LR(S_i) > LR(L_{LINE_NO}))$, (strip of type 10 containing horizontally overlapping lines) set $H(S_i) = H(S_i) - (LR(L_{LINE_NO}) - FR(L_{LINE_NO})$, $LINE_NO = LINE_NO + 1$. Also set $FR(L_{LINE_NO}) = LR(L_{LINE_NO - 1}) + 1$ and go to step 5.1(for the same strip).

Step 5.6: if $LR(S_{i+1}) \leq LR(L_{LINE_NO})$, set $i = i + 1$ (strip type 7 containing only lower zone). Repeat step 5.6 (for multiple strips containing only lower zone).

Step5.7: Set *LINE_NO* = *LINE_NO* + 1. $FR(L_{LINE_NO}) = LR(L_{LINE_NO - 1}) + 1$. Go to step 5(for next strip).

}

Step 6: For j = 1 to *LINE_NO*

Display $FR(L_j)$ to $LR(L_j)$ as line boundaries.

END

Figure 4.6: Line boundaries identified by applying Algorithm 4.1 on Figure 4.5a

Figure 4.6 shows the boundaries of lines identified using the proposed algorithm. This algorithm has shown a remarkable improvement in the accuracy, for segmenting the horizontally overlapping lines and associating the small strips to their respective lines. Furthermore, this algorithm works efficiently even if the input document contains more than two consecutive horizontally overlapping lines. In the document shown in Figure 4.7, there are three consecutive horizontally overlapping lines in strip numbered 1 and five consecutive horizontally overlapping lines in strip numbered 2. The proposed Algorithm 4.1 segments all these strips correctly into individual lines as shown in Figure 4.8.

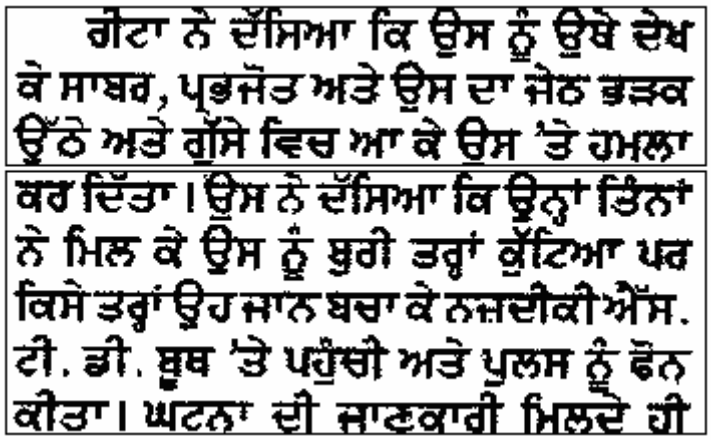


Figure 4.7: Strips containing horizontally overlapping lines

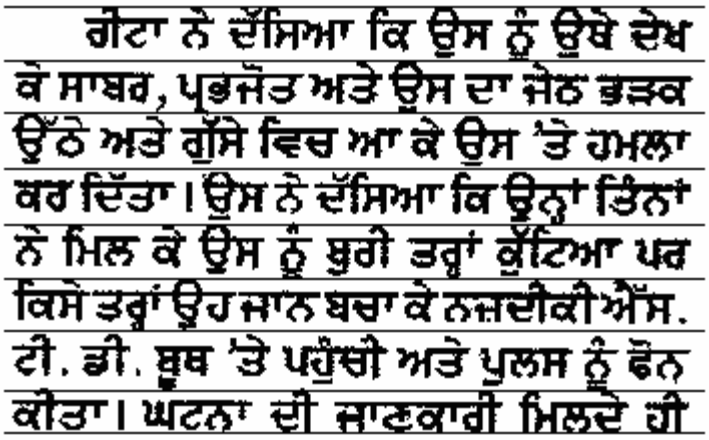


Figure 4.8: Line boundaries identified after applying Algorithm 4.1 on Figure 4.7

We have also carried out a statistical analysis of degraded printed Gurmukhi documents in order to find the percentage occurrence of 2, 3, 4 and 5 overlapping lines in a strip. Table 4.2 contains the results of this analysis. Most of the times, only two lines overlap but a total of 27.43% of strips (consisting of overlapping lines) contain more than two overlapping lines. This statistics is again based on the analysis of one hundred thirty four scanned documents from printed newspaper articles.

Table 4.2: Percentage of occurrence of strips containing two and more than two overlapping lines in newspaper documents

Number of overlapping lines	Percentage of occurrence
2	72.57
3	12.43
4	10.46
5	4.54

4.4.2 Segmentation of overlapping lines of uniform size in other Indian scripts

We had initially applied our algorithm of line segmentation on the documents of Gurmukhi script. Later on, we considered to apply this algorithm on other Indian scripts, since similar kinds of problems exist in these scripts also. We have identified the similar problems in seven other Indian scripts, namely, Devanagari, Bangla, Gujarati, Kannada, Tamil, Telugu and Malayalam. Figures 4.9a, 4.10a, 4.11a, 4.12a, 4.13a, 4.14a and 4.15a contain documents from Devanagari, Bangla, Gujarati, Kannada, Tamil, Telugu and Malayalam scripts, respectively, that have the problem of over/under line segmentation. These figures clearly show that the problem of overlapping lines (under segmentation) and the problem of broken components of lines (over segmentation) exist in these scripts also. As we noticed, documents of these scripts contain different types of strips out of total ten strip types identified for Gurmukhi script. Table 4.3 contains the data about the category of each strip from Figures 4.5a, 4.9a, 4.10a, 4.11a, 4.12a, 4.13a, 4.14a and 4.15a.

Table 4.3: Types of various strips of Figures 4.5a, 4.9a, 4.10a, 4.11a, 4.12a, 4.13a, 4.14a and 4.15a

Script / Strip type	Gurmukhi	Devanagari	Bangla	Gujarati	Kannada	Tamil	Telugu	Malaya-lam
Type 1	1			2				3
Type 2	2, 6							
Type 3	3	2		1				
Type 4	9				2		2	
Type 5	4		1, 3	4	1, 4, 6		1	1, 5
Type 6	5							
Type 7	7				3		6	
Type 8	8						3	
Type 9				3				4
Type 10	8	3,4	2	3	5	1,2	4	2

The statistical analysis of documents from these scripts reveals the information as shown in Table 4.4, about the percentage of occurrence of various types of strips.

Table 4.4: Percentage of various types of strips in Devanagari, Bangla, Gujarati, Kannada, Tamil, Telugu and Malayalam scripts

Script / Strip type	Devanagari	Bangla	Gujarati	Kannada	Tamil	Telugu	Malayalam
Type 1	0	0	13.34	0	8.69	0	2.70
Type 2	0	0	0	0	0	0	0
Type 3	49.56	12.65	0	9.67	0	8.72	0
Type 4	0	0	0	6.45	0	0	0
Type 5	8.1	83.23	40.00	61.31	30.43	65.21	72.98
Type 6	0	0	3.33	0	0	0	0
Type 7	0	0	0	9.67	0	4.34	0
Type 8	0	0	0	0	0	4.34	0
Type 9	0	0	3.33	0	8.69	0	0
Type 10	42.34	04.12	40.00	12.90	52.19	17.39	24.32

These results have been obtained by analyzing single column news articles from newspapers of corresponding scripts. We have considered 26 articles from Devanagari script, 18 articles from Bangla script, 29 articles from Gujarati script, 22 articles from Kannada script, 26 articles from Tamil script, 24 articles from Telugu script and 28 articles from Malayalam scripts in order to obtain these results. A zero entry in Table 4.4 indicates that we could not obtain this type of strip in available documents, but there remains a possibility of finding these types of strips as well. It can also be seen from Table 4.4 that the problem of horizontally overlapping lines in Devanagari and Tamil scripts is acute and it is moderate in other scripts.

We have proposed a new Algorithm 4.2 (**Segment_Lines_Scripts)** after modifying Algorithm 4.1 (**Segment_Lines).** Algorithm **Segment_Lines_Scripts** has been developed for segmenting horizontally overlapping lines and joining broken components of over segmented lines together in all eight Indian scripts including Gurmukhi script. This algorithm is first modified version of algorithm **Segment_Lines**. This algorithm segments a text document (of any above mentioned scripts) into lines.

This algorithm is developed on the basis of a heuristic that considers a relation between the height of upper zone, height of middle zone and height of lower zone. Based on our experimental observations, we consider two parameters **P1** and **P2** for the implementation of this heuristic. **P1** has been defined as the ratio of the height of upper zone to that of average line height (*AVG_LINE_HEIGHT*) and **P2** has been defined as the ratio of the height of lower zone to that of the height of middle zone. On the basis of experimental analysis on various documents of different scripts, we have adjusted the values of **P1** and **P2**, *e.g.*, for all headlines based scripts, the height of middle zone is approximately equal to the double the height of lower zone. Therefore, **P2** is set as 0.50 (height of lower zone/height of middle zone) for headline based scripts. Similarly, based on the values of ratio (height of lower zone/height of middle zone), the values of **P2** are set for different scripts.

Values of **P1** and **P2** are different for different scripts due to the structural properties of the lower and upper zone characters of these scripts. We have given the values of **P1** and **P2** for different scripts in Table 4.5.

The values of **P1** and **P2** have been set in accordance with the available sample data. These values can be adjusted to more accurate values in accordance with the different kinds

of sample documents available in different scripts.

Table 4.5: Value of P1 and P2 for different scripts

Script	Value of P1	Value of P2
Gurmukhi	0.30	0.50
Devanagari	0.30	0.50
Bangla	0.30	0.50
Gujarati	0.30	0.50
Kannada	0.25	0.70
Tamil	0.30	0.70
Telugu	0.40	0.80
Malayalam	0.30	0.70

Algorithm 4.2a contains the outline of algorithm **Segment_Lines_Scripts** while Algorithm 4.2b contains the detailed algorithm

Algorithm 4.2a: Segment_Lines_Scripts (document matrix in binary form)

begin

get binary matrix of the input document;
find all strips and compute the height of each strip;
if (document is from headline based script)
find the positions of all headlines;
else
find the positions of all meanlines;
end-if
compute average line height from the position of headlines/meanlines;
for (all strips)
if (height of strip is less than **P1** × of average line height)
strip belongs to type 1, do nothing and loop for next strip;
end-if
if (height of strip is greater than 50% of average line height)

strip belongs to type 2, 3, 4, 5, 6, 8, 9 or 10;
compute baseline position;
compute height of the middle zone from headline and baseline positions;
add (P2 × height of middle zone) to baseline row for finding actual line boundary (type 2, 3, 4, 5, 6, 8 and 9 solved);
if (strip boundary is greater than actual line boundary)
strip belongs to type 10;
decrease strip height by one complete line;
loop for the same strip;
end-if
if (strip boundary is less than actual line boundary)
strip belongs to type 7
repeat
consider next strip;
until (strip boundary is less than actual line boundary)
end-if
end-if
end-for
end-algorithm

Algorithm 4.2b (Segment_Lines_Scripts): segmentation of uniform sized text lines of printed Indian scripts

BEGIN

Step 1: Using the horizontal projections, different strips in input binary document are identified. For that, whenever $HP(i) = 0$ for $i = 1, 2, 3, \ldots, L$, it is marked as the boundary of strip line. Let us denote the strips by $S_1, S_2, S_3, \ldots, S_m$ and first row of strip p as $FR(S_p)$, last row of strip p as $LR(S_p)$. Height of the strip is calculated as $H(S_p)=LR(S_p)-FR(S_p) + 1$, for $p = 1, 2, 3, \ldots, m$. Strips identified in documents from eight Indian scripts are shown in Figures 4.5a, 4.9a, 4.10a, 4.11a, 4.12a, 4.13a, 4.14a and 4.15a.

Step 2: If input document is from any headline based script, go to step 3 else go to step 4.

Step 3: Identify the position of headlines using horizontal projections. Denote the ending position of the headlines as $H_1, H_2, H_3, \ldots, H_n$. Also denote the lines to be identified as $L_1, L_2, L_3, \ldots, L_n$.

//number of headlines are same as number of actual lines

Go to step 5.

Step 4: Identify the position of meanlines using first order differences of horizontal projection. Indicate the position of the meanlines as $H_1, H_2, H_3, \ldots, H_n$.

//number of meanlines are same as number of actual lines.

Step 5: Define

$$AVG_LINE_HEIGHT = \frac{1}{n-1}\sum_{i=2}^{n}(H_i - H_{i-1})$$

Step 6: Set *LINE_NO* = 1 and first row of line *LINE_NO* as first row of first strip, *i.e.*, $FR(L_{LINE_NO}) = FR(S_1)$.

Step 7: For $i = 1$ to m, perform the following operations:

Step 7.1: if ($H(S_i) <$ (**P1** $\times$ *AVG_LINE_HEIGHT*)), S_i is of type 1.

//contains only upper zone

Repeat step 7.

//ignore current strip and go for next strip.

Step 7.2: if ($H(S_i) >$ (0.50 $\times$ *AVG_LINE_ HEIGHT*)), S_i will be of type 2, 3, 4, 5, 6, 8, 9 or 10 and will contain at least one headline/meanline and one baseline.

Step 7.3: identify the position of baseline. Mark it as $BASE_{LINE_NO}$. Also set height of the middle zone as $HGT_MID = BASE_{LINE_NO} - H_{LINE_NO}$.

Step 7.4: set last row of line *LINE_NO* as $LR(L_{LINE_NO}) = BASE_{LINE_NO} +$ **P2** $\times$ (*HGT_MID*).

//This will solve the segmentation problem of strip type 2, 3, 4, 5, 6, 8 and 9.

Step 7.5: if ($LR(S_i) > LR(L_{LINE_NO})$)

//strip type 10 containing horizontally overlapping lines

set $H(S_i) = H(S_i) - (LR(L_{LINE_NO}) - FR(L_{LINE_NO}))$, increment *LINE_NO* by 1. Also set $FR(L_{LINE_NO}) = LR(L_{LINE_NO-1}) + 1$ and go to step 7.1.

//for the same strip.

Step 7.6: if $(LR(S_{i+1}) \leq LR(L_{LINE_NO}))$, set $i = i + 1$.

//strip type 7 containing only lower zone

Repeat step 7.6.

//for multiple lower zones.

Step 7.7: increment *LINE_NO* by 1. Set $FR(L_{LINE_NO}) = LR(L_{LINE_NO-1}) + 1$. Go to step 7.

//for next strip.

Step 8: for $j = 1$ to *LINE_NO*

Display $FR(L_j)$ to $LR(L_j)$ as line boundaries.

END.

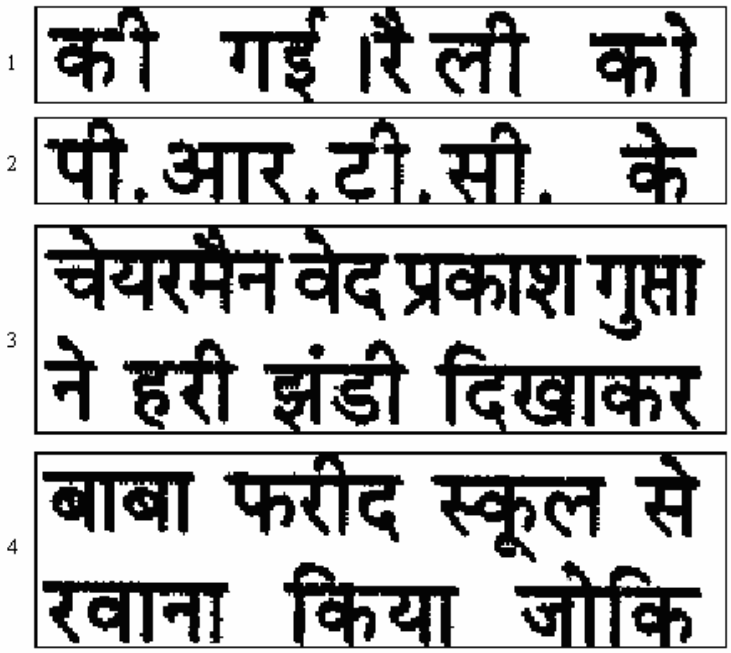


Figure 4.9a: Strips in printed Devanagari script document

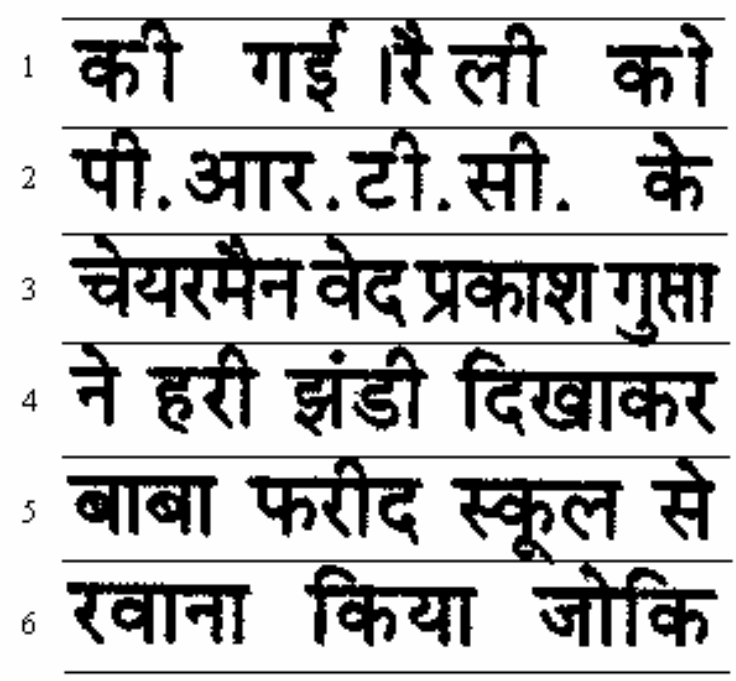


Figure 4.9b: Line boundaries identified using proposed Algorithm 4.2

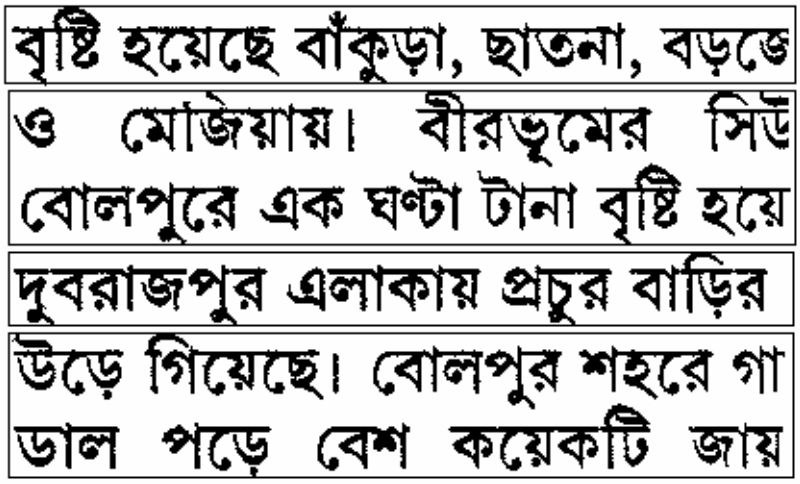


Figure 4.10a: Strips in printed Bangla script document

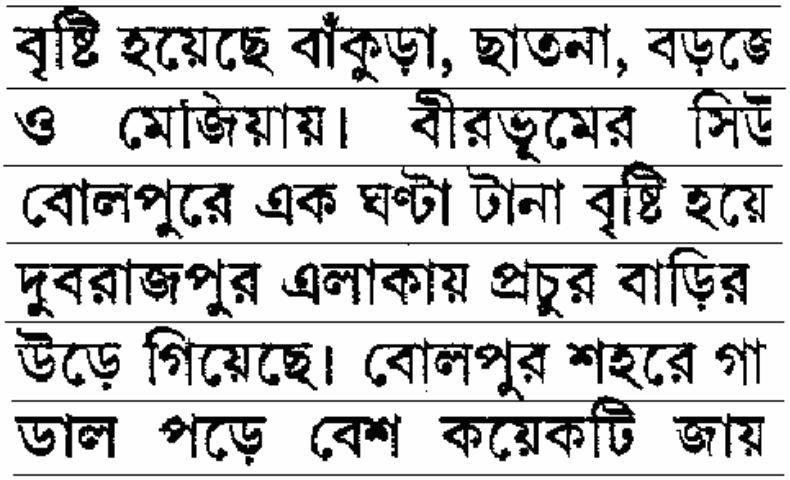


Figure 4.10b: Line boundaries identified using proposed Algorithm 4.2

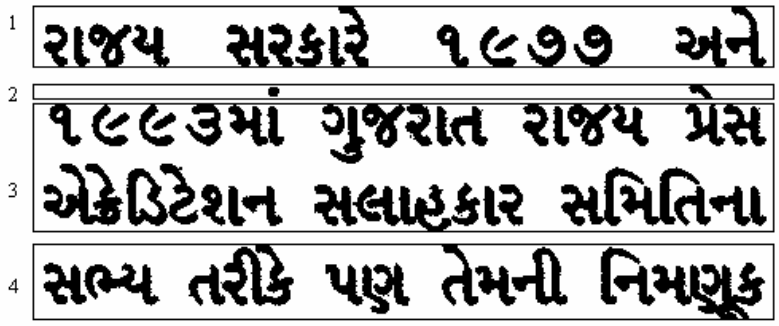


Figure 4.11a: Strips in printed Gujarati script document

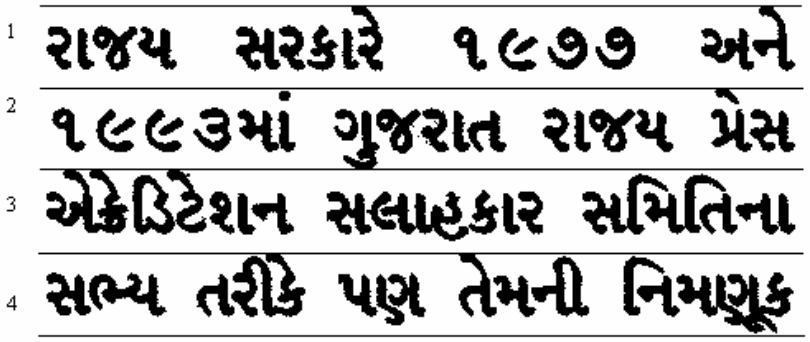


Figure 4.11b: Line boundaries identified using proposed Algorithm 4.2

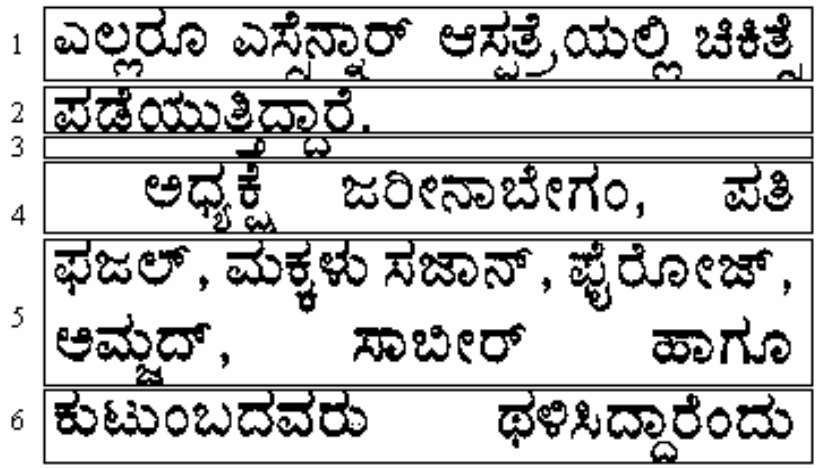


Figure 4.12a: Strips in printed Kannada script document

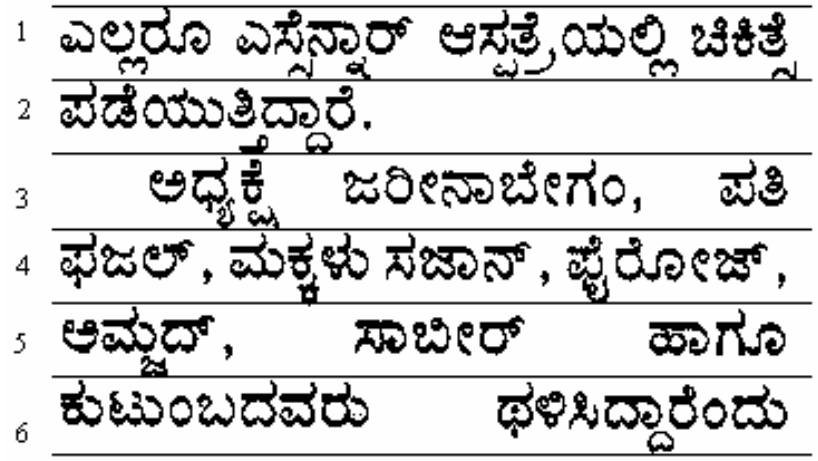


Figure 4.12b: Line boundaries identified using proposed Algorithm 4.2

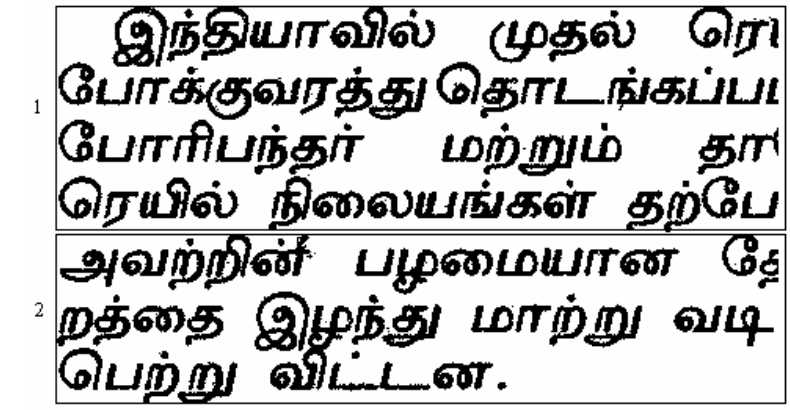


Figure 4.13a: Strips in printed Tamil script document

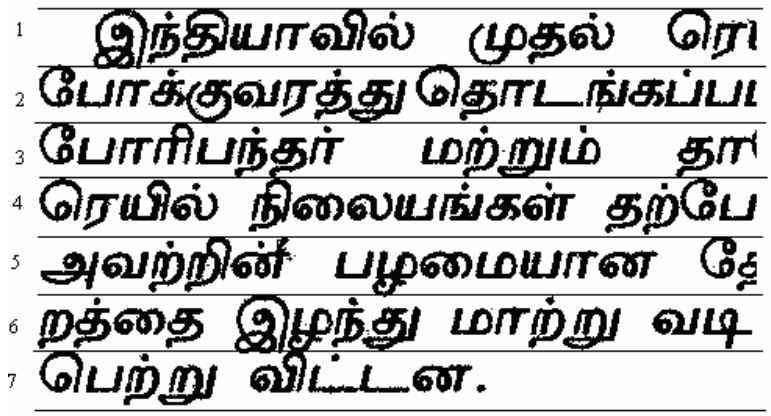


Figure 4.13b: Line boundaries identified using proposed Algorithm 4.2

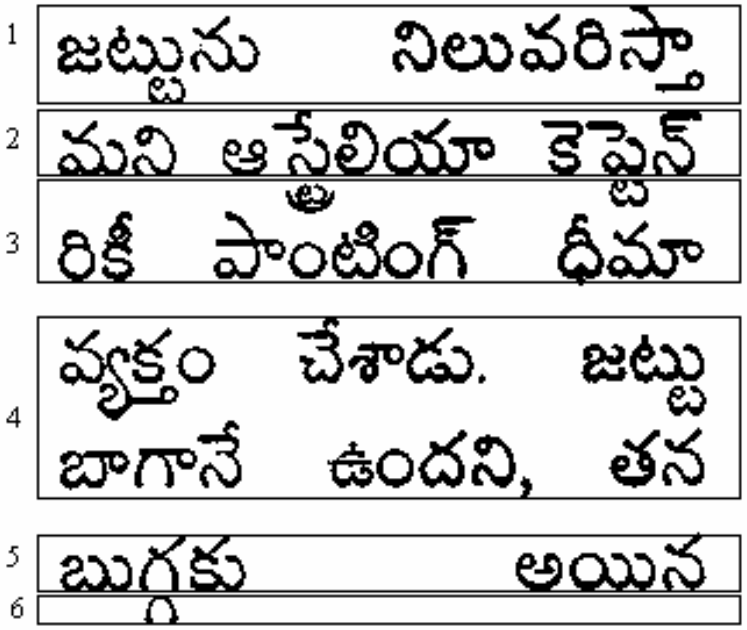


Figure 4.14a: Strips in printed Telugu script document

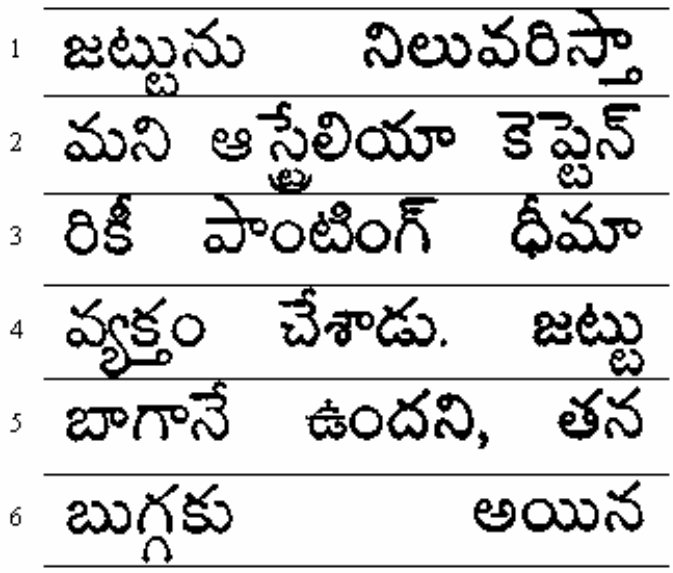


Figure 4.14b: Line boundaries identified using proposed Algorithm 4.2

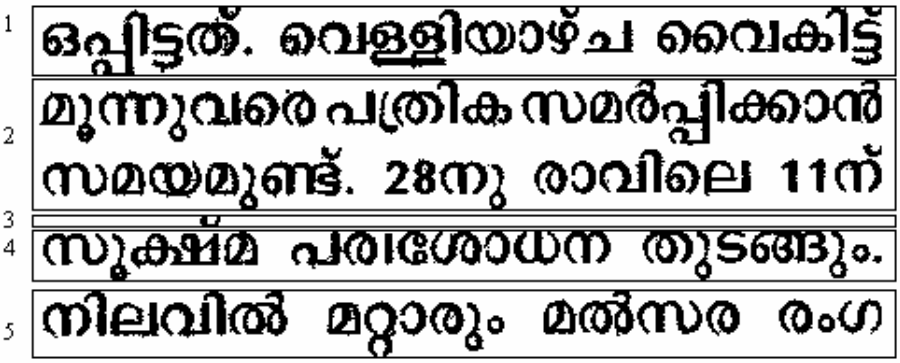


Figure 4.15a: Strips in printed Malayalam script document

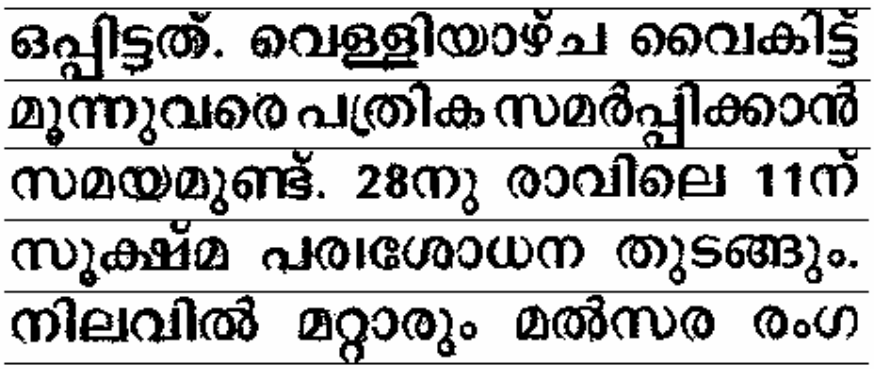


Figure 4.15b: Line boundaries identified using proposed Algorithm 4.2

Figures 4.9b, 4.10b, 4.11b, 4.12b, 4.13b, 4.14b and 4.15b show the boundaries of lines identified in the documents of different scripts using the proposed Algorithm 4.2.

In order to find the position of headline using Algorithm 4.2, for headline-based scripts, one can use any standard technique [10, 70, 76, 77]. Similarly, for non-headline based scripts, standard technique proposed by Dholakia *et al.* [75] can be used. We have used the following code for identification of headline/meanline.

Step headline/meanline:

if (script = headline based)

find $MAXPIX = \max\{HP(i)\}$, for $i = 1, 2, 3, \ldots, L$. The headlines are considered as those lines whose $HP(i) \geq 70\%$ of $MAXPIX$;

else

find first order differences of horizontal projections, *i.e.*, $dx[i] = HP[i + 1] - HP[i]$;

if (script = Gujarati)

while ($dx[i] > 0$)

increment *i by* 1*;*

end-while

start = *i*;

while (dx[i] ≤ 0)

increment i *by* 1*;*

end-while

end = *i*;

for (*j* = *end* to (*end* + (*end* − *start*)))

find max(*HP*[*j*]);

end-for

row number *j* is meanline called *mean*.

upper = *mean* − *start;*

else

for (*i* = 1 to *L*)

find *maxh* = max(*HP*[ ]);

end-for

while ($dx[i] < (0.55 \times maxh)$)

increment *i by* 1;

end-while

i marks the meanline.

end-if

For finding the position of baseline, for headline-based scripts, one can again use any standard technique [10, 70, 76, 77]. We have used the following code for identification of baseline for both headline based and non-headline based scripts.

Step baseline:

if (script = headline based)

identify the position of baseline by noting the continuous vertical projection *CVP(k)*, $\{k = H_{LINE_NO}$ to $LR(S_i)\}$. The position where *CVP(k)* ends, mark it as α every time. The row in which maximum α are found is considered to be the baseline;

else

find first order differences of horizontal projections, *i.e.*, $dx[i] = HP[i + 1] - HP[i]$;

if (script = Gujarati)

for ($j = (mean + (upper \times 1.5))$ to $(mean + (upper \times 2.5))$)

find min($HP[j]$);

end-for

row number *j* is baseline;

else

for (i = 1 to L)

find $minh = \min(HP[i])$;

end-for

while ($dx[i] > (0.55 \times minh)$)

increment *i by* 1;

end-while

i marks the baseline.

end-if

The algorithm **Segment_Lines_Scripts** has shown a remarkable improvement in the accuracy for segmenting the horizontally overlapping lines and associating the broken components of a line to their respective lines. This algorithm works even if the input document contains many consecutive horizontally overlapping lines. As shown in Figure 4.13a, there are four consecutive horizontally overlapping lines in strip number 1 and three consecutive horizontally overlapping lines in strip number 2. The proposed Algorithm 4.2 segments all the overlapping lines of these strips correctly into individual lines as shown in Figure 4.13b.

These kinds of problems have been found in newspapers of almost all printed Indian scripts. We have analyzed and solved the problem in eight most widely used printed Indian scripts, namely, Gurmukhi, Devanagari, Bangla, Gujarati, Kannada, Tamil, Telugu and Malayalam. Besides this, the same algorithm can be used for segmenting overlapping lines in Oriya script.

The algorithm **Segment_Lines_Scripts** fails in case of Urdu script. As shown in Figure 4.16, the problem of horizontally overlapping lines also exists in Urdu script. As shown in Figure 4.16, strip number 1 contains two and strip number 2 contains four horizontally overlapping lines. There is no concept of three zones, the position of headline/meanline and the position of baseline in Urdu script. Since, algorithm **Segment_Lines_Scripts** is based upon the concept of three zones, the position of headline/meanline and the position of baseline, it will not segment horizontally overlapping lines of Urdu script.

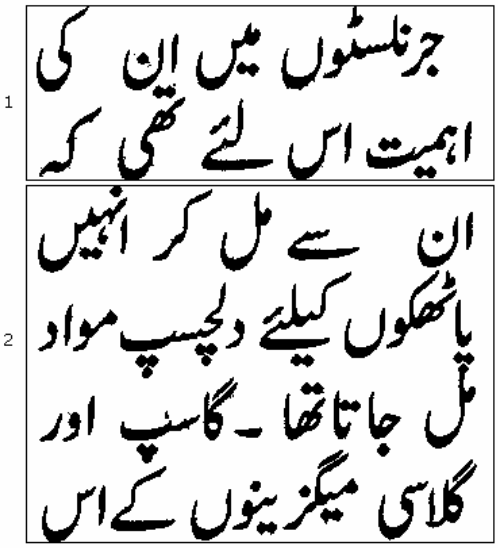


Figure 4.16: Urdu script document containing overlapping lines

4.5 Segmentation of Overlapping Lines of Different Sized Text

Documents may contain the text with a large variation in text size. The heading text lines of newspaper document are always larger in size than the actual news text size. One can infer from Figure 4.17a that first two lines that are the heading of the news have larger text size (let us call it segment 1) than the text size of the news text (let us call it segment 2). The Algorithm 4.2 works accurately for segmenting overlapping lines of the uniform text size. On the contrary, when two different texts sized lines overlap the algorithm does not work that accurately. Figure 4.17b contains the output of the Algorithm 4.2 when applied on the document shown in Figure 4.17a. One can see that line number 2 and 3 are not properly segmented. Line number 2 has overlapped with line number 3, thus producing incorrect segmentation by breaking some text portion of the third line and adding it to the second line. Other lines have been correctly segmented. As such, two consecutive lines having different text size are not properly segmented.

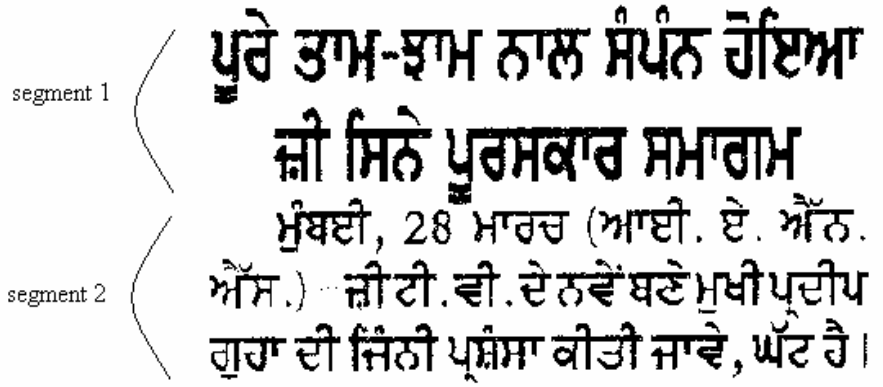


Figure 4.17a: Printed text document of Gurmukhi script containing different font size text

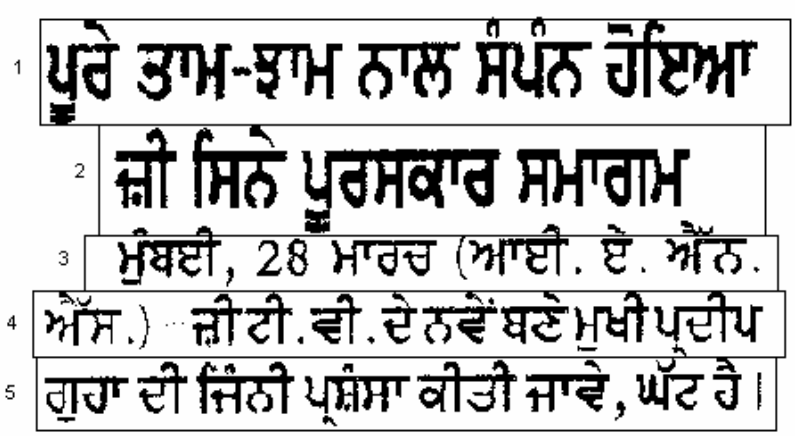


Figure 4.17b: Incorrect line boundaries of Figure 4.17a, identified using Algorithm 4.2

We have again modified Algorithm 4.1 (**Segment_Lines)** in order to solve the problem of segmenting horizontally overlapping lines of different sized text. Based on observations, it has been found that the number of lines in segment 1 is always less than the number of lines in segment 2, for almost all documents (news items). As such, the value of *AVG_LINE_HEIGHT* will be closer to average line height of segment 2. We have proposed Algorithm 4.3 (**Segment_Lines_Diff_Sized)** by modifying step 7 of Algorithm 4.1**.** Rests of the steps of algorithm **Segment_Lines_Diff_Sized** are same as that of algorithm **Segment_Lines.**

Algorithm 4.3 (Segment_Lines_Diff_Sized): segmentation of different sized text lines of degraded printed Gurmukhi script

Step 7: For $i = 1$ to m perform the following operations:

Step 7.1: //for strip from segment 1

if $((H_{LINE_No+1} - H_{LINE_No}) > (1.4 \times (AVG_LINE_HEIGHT)))$

$PT = 0.35$, $PM = 0.60$ goto step 7.4.

Step 7.2: //for strip joining segment 1 and segment 2

if $((H_{LINE_No+1} - H_{LINE_No}) > (1.15 \times (AVG_LINE_HEIGHT)))$

$PT = 0.35$, $PM = 0.60$, $flag = 1$, go to step 7.4.

Step 7.3: //for strip from segment 2

if $((H_{LINE_No+1} - H_{LINE_No}) < (AVG_LINE_HEIGHT))$ set $flag1 = 1$ and

$PT = 0.25$, $PM = 0.40$.

Step 7.4: if ($H(S_i) < (PT \times AVG_LINE_HEIGHT)$), S_i is of type 1.

//contains only upper zone

Repeat step 7.

//ignore current strip and go for next strip.

Step 7.5: if ($H(S_i) > (PM \times AVG_LINE_HEIGHT)$), S_i will be of type 2, 3, 4, 5, 6, 8, 9 or 10 and will contain at least one headline/meanline and one baseline.

Step 7.6: identify the position of baseline. Mark it as $BASE_{LINE_NO}$. Also set height of the middle zone as $HGT_MID = BASE_{LINE_NO} - H_{LINE_NO}$.

Step 7.7: set last row of line *LINE_NO* as $LR(L_{LINE_NO}) = BASE_{LINE_NO} + (0.50 \times (HGT_MID)$.

// this will solve the segmentation problem of strip type 2, 3, 4, 5, 6, 8 and 9.

Step 7.7.1: if (*flag* = 1 and *flag1* = 1) adjust $FR(L_{LINE_NO}) = SH_i - (0.50 \times (HGT_MID)$

//as shown in Figure 4.18, we have adjusted the starting row of 3rd line which was incorrectly segmented by Algorithm 4.1

Step 7.8: if ($LR(S_i) > LR(L_{LINE_NO})$)

// strip type 10 containing horizontally overlapping lines

set $H(S_i) = H(S_i) - (LR(L_{LINE_NO}) - FR(L_{LINE_NO})$, increment *LINE_NO* by 1, set $FR(L_{LINE_NO}) = LR(L_{LINE_NO-1}) + 1$ and go to step 7.1.

//for the same strip.

Step 7.9: if ($LR(S_{i+1}) \leq LR(L_{LINE_NO})$) increment *i* by 1

// strip type 7 containing only lower zone

Repeat step 7.6.

//for multiple lower zones.

Step 7.10: increment *LINE_NO* by 1, set $FR(L_{LINE_NO}) = LR(L_{LINE_NO-1}) + 1$, go to step 7.

//for next strip.

As shown in Figure 4.18, Algorithm 4.3 has segmented lines numbered 2 and 3 in such a way that line number 2 contains some part of line number 3 which can be considered as noise for line number 2. We can simply remove this noise on the basis of width of the stroke or location of the noise as it is at the bottom of the line or by considering these as isolated connected components at the bottom of the line. The problem of line number 3 has been solved, as complete upper zone of this line has been retained in it. Unfortunately, line number 3 also contains some part of line number 2 which can again be considered as noise for line number 3. We can remove this noise also on the basis of width of the stroke or location of the noise as it is at the top of the line or by considering these as isolated connected components at the top of the line.

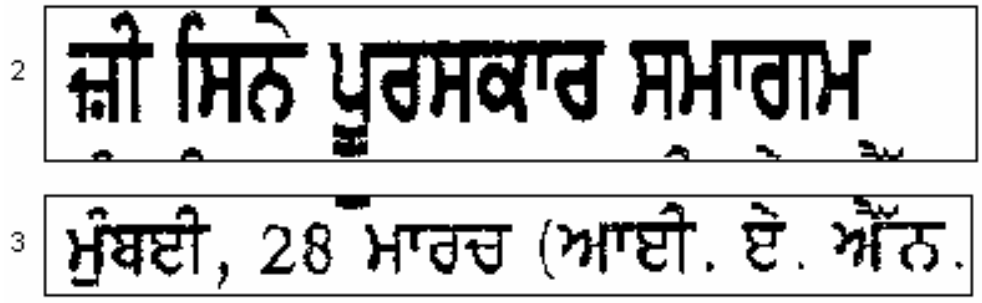


Figure 4.18: Segmented lines, numbered 2 and 3, of Figure 4.17a using Algorithm 4.3

4.6 Word and Zone Segmentation

Segmentation of a line in words is the next step in segmentation phase. Generally, there is sufficient amount of space between words, even in degraded documents. Due to this observation, word segmentation does not remain a complex problem to solve. In this thesis, we have used inter word gap for word segmentation. We have also used the same technique as used by Chaudhuri and Pal [76] for segmenting the words using vertical projection. If two or less black pixels are encountered in vertical scan, then the scan is denoted by 0, else the scan is denoted by the number of black pixels. In this way, a vertical projection profile is constructed as shown in Figure 4.19. Now, if in the vertical projection profile there exists a run of at least k consecutive zeros then the midpoint of that run is considered as the boundary of a word. Generally, the value of k is taken as half of the text line height. As shown in

Figure 4.19, there is a sufficient amount of gap in horizontal direction in the vertical projection of a line of degraded Gurmukhi script for segmenting the words.

Figure 4.19: Word segmentation using vertical projection

After segmenting the words, zone segmentation has been carried out. Zone segmentation includes identification of upper, middle and lower zone boundaries. The process of finding headlines and baselines has already been explained in Algorithms 4.1 and 4.2. As already discussed, the region above the headline is called upper zone, the region from the headline to the baseline is called middle zone and region below the baseline is called lower zone. Figure 4.20 contains all three segmented zones of one word.

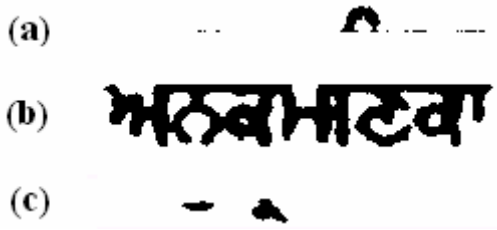


Figure 4.20: Zone segmentation: (a) upper zone, (b) middle zone, (c) lower zone

4.7 Character Segmentation

Once the line and word segmentation has been achieved, and words have been segmented into zones, we have to segment the characters for the purpose of recognising them. Character segmentation is an extremely important step in a text recognition system and the accuracy of the text recognition system heavily depends on this step. The importance of character segmentation process increases, when the input document is degraded.

Segmentation of text into zones breaks up those characters into two components which are spread into two zones. Also, few characters are multicomponent characters that consist of two disconnected components. Therefore, from here onwards we have called each symbol in upper, middle and lower zone as sub-symbol (connected component, *i.e.*, it may be a consonant, vowel bearer, additional consonant, vowel modifier, auxiliary signs half character, broken component or a component of a multicomponent character). As such, character segmentation can also be termed as sub-symbol segmentation now. Few symbols which break into subsymbols have been shown in Table 4.6. Besides these, all additional consonants break into two components.

Table 4.6: Sub-symbols of multicomponent characters and characters spanned to two zones in Gurmukhi script

Symbol	Sub-symbols	Symbol	Sub-symbols
ਉ	੍ਹ and ਓ	ਿ	ਾ and ੰ
ਗ	ਹ and ਾ	ੀ	ਾ and ੰ
=	਼ and ਼	ੳ	੍ਹ and ਓ

After that the conjuncts and composite characters have been segmented by composite character segmentation.

All the techniques used in literature do not work well when the document contains touching characters. Special care is needed to segment touching characters. We have developed some new algorithms for segmenting touching characters of Gurmukhi script in various zones. First, we have categorized touching characters in all three zones of printed degraded Gurmukhi script. Next, the strategies for segmenting touching characters have been proposed for each zone.

4.7.1 Categories of touching sub-symbols in upper zone

The sub-symbols in upper zone usually consist of vowel modifiers, auxiliary signs and stroke of vowel bearer. We can divide these sub-symbols into following three categories:

Category 1: sub-symbols consisting of vowels present in upper zone only.
Category 2: sub-symbols consisting of a stroke present in upper zone of a symbol spanned to middle and upper zones.
Category 3: sub-symbol consisting of a stroke of a vowel bearer in upper zone.

In the first category, whole of the vowel modifier or auxiliary sign lies in upper zone. In second category, there are two vowel modifiers ੀ and ਿ, whose one stroke lies in upper zone and produces touching patterns with other sub-symbols in upper zone. Third category contains other shapes appearing in upper zone. For example, vowel bearer ੲ contains a stroke ੍ which lies in upper zone and may touch with other sub-symbols of upper zone. Similarly, when vowel bearer ੲ appears with vowel modifier *hōrā*, it changes the shape of *hōrā* and results in combined shape of ਓ. The sub-symbol ੁ present in upper zone of symbol ਓ may touch with other sub-symbols in upper zone. Table 4.7 consists of pronunciation of name, category and shape of the vowels, auxiliary signs and other symbols containing sub-symbols in upper zone.

Table 4.7: Pronunciation of name, category and actual shape of vowels, auxiliary signs and other symbols containing sub-symbols in upper zone

Name of the symbol	Category	Shape of the symbol
sihārī	second	ਿ
bihārī	second	ੀ
lānvām	first	ਾ
dulānvām	first	ੈ
hōrā	first	ੋ
kanaurā	first	ੌ
adhak	first	ੱ
bindī	first	ਂ
tippī	first	ੰ
ūrā	third	ੂ
hōrā	third	ੁ

The pronunciation, actual shape and examples of the sub-symbol falling in first category are shown in Figure 4.21, falling in second category are shown in Figure 4.22 and falling in third category are shown in Figure 4.23.

Figure 4.21: Pronunciation of name, actual shape and example words of the vowels falling in first category

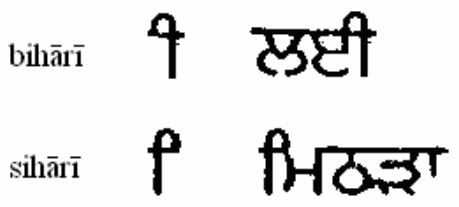


Figure 4.22: Pronunciation of name, actual shape and example words of the vowels containing sub-symbols in upper zone, falling in second category

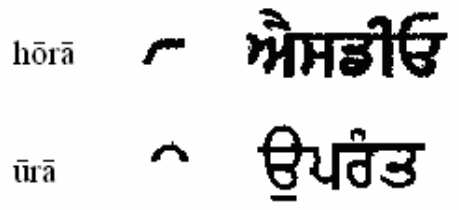


Figure 4.23: Pronunciation of name, actual shape and example words of Gurmukhi characters having sub-symbols strokes in upper zone falling in third category

On the basis of observations about the sub-symbols in upper zone, the following three categories have been proposed in the upper zone for touching sub-symbols.

Category a.1: *bindī* (◌ं) touching with other sub-symbols

By carefully analyzing the Gurmukhi documents, it is found that 35% of the total pair of touching sub-symbols in upper zone fall in this category. In this category, vowel *bindī* (dot shaped) touches with other sub-symbols present in upper zone either from left or right side. *Bindī* is used with bihārī, lānvām, dulānvām, kanaurā, hōrā and ūrā in upper zone. Figure 4.24(a) contains words from Gurmukhi script in which *bindī* touches with other sub-symbols in upper zone.

Category a.2: *adhak* (◌ॅ) touching with other sub-symbols

Approximately, 52% of touching sub-symbols of the total touching sub-symbols in upper zone fall in this category. In this category, *adhak vowel* touches with other sub-symbols present in upper zone. Figure 4.24(b) contains some examples of *adhak* touching with other sub-symbols in upper zone.

Category a.3: *tippī* (ੰ) touching with other sub-symbols

We have noticed that 13% of touching sub-symbols of the total touching sub-symbols in upper zone fall in this category. In this category, *tippī* vowel touches with other sub-symbols present in upper zone. Furthermore, it has been revealed from the analysis that the vowel *tippī* always touches with upper zone segment of the vowel *sihārī* (ਿ). Figure 4.24(c) contains examples of *tippi* touching with upper zone segment of *sihārī* in upper zone.

Figure 4.24: Gurmukhi words containing touching sub-symbols in upper zone (touching sub-symbols have been marked with circles), (a) *bindī* touching with other sub-symbols, (b) *adhak* touching with other sub-symbols, (c) *tippī* touching with other sub-symbols

Table 4.8 contains some of the properties of categories of touching sub-symbols in upper zone.

Table 4.8: Properties of touching sub-symbol categories in upper zone

Category Number	Percentage of touching sub-symbols of the category	Structural features of the category	sub-symbols of the category
a.1	35	Dot shaped	*bindī* (ਂ)
a.2	52	Concaved shaped	*adhak* (ੱ)
a.3	13	Convexed shaped	*tippī* (ੰ)

4.7.2 Categories of touching sub-symbols in middle zone

The sub-symbols in middle zone usually consist of the following consonants, vowel bearers, the additional consonants, vowel and stroke of vowel.

The consonants: ਸ ਹ ਕ ਖ ਗ ਘ ਙ ਚ ਛ ਜ ਝ ਞ ਟ ਠ ਡ ਢ ਣ ਤ ਥ ਦ ਧ ਨ ਪ ਫ ਬ ਭ ਮ ਯ ਰ ਲ ਵ ੜ

The vowel bearers: ੳ(ੳ) ਅ ੲ

The additional consonants (multi component characters): ਸ਼ ਜ਼ ਖ਼ ਫ਼ ਗ਼ ਲ਼

Vowels and stroke of vowel: ਾ ਿ(ੀ) ੁ(ੂ)

After carefully analyzing the database of touching sub-symbols in middle zone, it is found that on the basis of structural properties of the Gurmukhi script, various touching sub-symbols can be classified into five categories. Some characters may fall in multiple categories. For each pair of touching sub-symbols, these categories are defined on the basis of left sub-symbol of the pair. These categories are hereby briefly described.

Category b.1: Touching sub-symbols containing full sidebar at right end

On the basis of statistical analysis, it is found that approximately 54% of the pairs of touching sub-symbols contain those characters at left side which have full sidebar (vertical straight line) at their extreme right end. There are total twelve consonants and one stroke of two vowels in Gurmukhi script containing sidebars at right end, as mentioned below.

ਅ, ਸ, ਖ, ਗ, ਘ, ਜ, ਥ, ਧ, ਪ, ਬ, ਮ, ਯ, ।

For example, as shown in Figure 4.25, touching sub-symbols at positions marked as 2, 3, 5, 6, 8, 11, 15, 16 and 17 fall in this category.

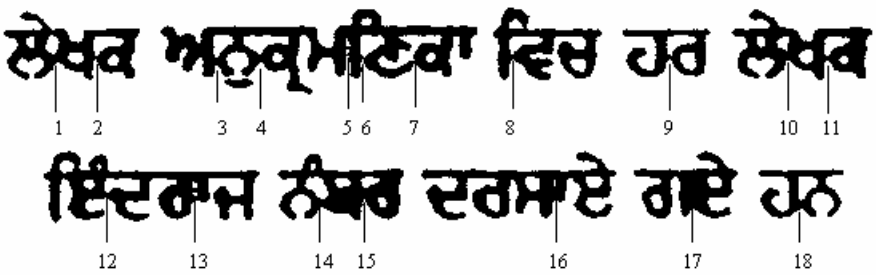


Figure 4.25: Words containing touching characters in middle zone

Category b.2: Touching sub-symbols containing partial sidebar at right end

There are four consonants in Gurmukhi script falling in middle zone that do not have full sidebar at their extreme right end, but it contains 75-85% of the full sidebar. Also, it has been noted that those sub-symbols have hook shaped feature at their right bottom. It has been observed that approximately 15% of touching sub-symbols fall in this category. These sub-symbols are: ਗ, ਰ, ਹ, ਚ. In Figure 4.25, touching sub-symbols at positions marked as 9, 13 and 18 belong to this category.

Category b.3: Touching sub-symbols containing little sidebar at right end

It has been observed that approximately 11% of touching sub-symbols fall in this category. In this category, the sub-symbols contain a little sidebar at the right side of the character. The approximate size of the sidebar is half of the height of middle zone. There are eight consonants and one vowel in Gurmukhi script falling in this category and these are: ਏ, ਛ, ਢ, ਟ, ਢ, ਦ, ਫ, ਵ, ਾ. Figure 4.25 contains touching sub-symbol at position numbered 12 from this category.

Category b.4: Touching sub-symbols containing curved shape at right end

It has been revealed from the analysis that approximately 16% of touching sub-symbols fall in this category. Here, touching sub-symbol contains curved shape at right extreme end. There are ten consonants in Gurmukhi script, namely: ਙ, ਝ, ਠ, ਡ, ਤ, ਲ, ਭ, ੜ, ੳ, ਨ falling in this category. Figure 4.25 contains touching sub-symbols at positions 1, 4, 10 and 14 from this category.

Category b.5: Touching sub-symbols of miscellaneous type

Rest, 4% of the total touching sub-symbols fall in neither of the above mentioned categories. Therefore, these sub-symbols are categorized in this category containing touching sub-symbols of miscellaneous type. There are two consonants in Gurmukhi script, namely: ਕ, ਣ

falling in this category. Figure 4.25 contains touching sub-symbol at position 7 from this category.

Table 4.9 contains the information about the five categories of touching sub-symbols in middle zone.

Table 4.9: Properties of touching sub-symbol categories in middle zone

Category number	Percentage of touching sub-symbols falling in the category	Structural features of the category	Shape of sub-symbols	Position numbers in Figure 4.25 containing touching pattern of this category
b.1	54	Full sidebar at right end	ਅ, ਸ, ਖ, ਗ, ਘ, ਜ, ਥ, ਧ, ਪ, ਬ, ਮ, ਯ, ।	2, 3, 5, 6, 8, 11, 15, 16, 17
b.2	15	Quarter side bar at right end	ਗ, ਰ, ਹ, ਚ	9, 13, 18
b.3	11	Half sidebar at right end	ਏ, ਟ, ਦ, ਵ, ਛ, ਵ, ਫ, ਾ	12
b.4	16	Curved shape at right end	ਙ, ਝ, ਠ, ਡ, ਤ, ਲ, ਭ, ੜ, ਉ, ਨ	1, 4, 10, 14
b.5	4	Miscellaneous sub-symbols	ਕ, ਣ	7

4.7.3 Categories of touching sub-symbols in lower zone

The sub-symbols in lower zone usually consist of the following vowels and half characters.

Vowel Modifiers: ◌ ◌

Half Characters: ◌ ◌ ◌

Based on the analysis, we have divided these sub-symbols in the following three categories in lower zone.

Category c.1: Lower zone sub-symbols touching with middle zone sub-symbols

Depending upon the quality of the input document, approximately 60% of the total lower zone vowels and half characters always touch the middle zone sub-symbols. This may sometimes happen even with non-degraded texts. Figure 4.26(a) shows Gurmukhi words containing this kind of touching sub-symbols.

Figure 4.26: Touching sub-symbols in lower zone (touching sub-symbols have been marked with circles), (a) lower zone sub-symbols touching with upper zone sub-symbols, (b) lower zone sub-symbols touching with each other, (c) two components of multi component vowel modifier ੍ਰ touching with each other

Category c.2: Lower zone vowels and half characters touching with each other

There is a possibility, though rare, of lower zone vowels touching with each other. Figure 4.26(b) shows this kind of touching pattern in lower zone.

Category c.3: Components of a vowel touching with each other

We have proposed category c.3, in which two components of a multi component vowel modifier (੍ਰ) touch with each other. Figure 4.26(c) consists of words containing this kind of touching pattern. Also it is analyzed that whenever ੍ਰ is present in Gurmukhi document, approximately in 70% of the cases, the two component of this vowel modifier always touch each other.

Table 4.10 contains the information about the three categories of touching sub-symbols in lower zone.

Table 4.10: Properties of touching sub-symbol categories in lower zone

Category number	Percentage of occurrence of touching pattern out of total occurrence of these sub-symbols	Characteristics of pattern
c.1	40-70	vowel modifiers and half characters touching with middle zone characters
c.2	0-1	vowel modifiers and half characters touching with each other
c.3	50-70	components of multicomponent vowel modifier touching with each other

We now propose algorithms for segmenting touching sub-symbols in three zones.

4.8 Segmentation of Touching Sub-symbols in Upper Zone

For segmenting touching sub-symbols in upper zone, we have proposed Algorithm 4.4 (**Segment_Upper)** based on the structural properties of Gurmukhi sub-symbols. Structural properties of Gurmukhi sub-symbols reveal that every sub-symbol in upper zone consists of single concavity or convexity in its structure. The concept of single concavity or convexity is used to segment the touching sub-symbols in upper zone. For detection of touching position, if a sub-symbol has a concavity followed a convexity or vice-versa; it is supposed to be the merged sub-symbols, *i.e.*, candidate of segmentation. In the merged sub-symbols, whenever the first concavity (or convexity) terminates, we have put a segmentation column to segment the touching sub-symbols in upper zone. Algorithm 4.4a contains outline of the algorithm **Segment _Upper,** while Algorithm 4.4b contains the detailed algorithm.

Algorithm 4.4a: Segment _Upper (binary matrix of sub-symbol in upper zone)

begin

get binary matrix of the sub-symbol in upper zone;

compute top profile of sub-symbol;

```
    for (all columns of sub-symbol)
        if (top profile moves upward)
            concavity confirmed;
            while (profile moves upward)
                    move to next column;
            end-while
            while (profile moves downward)
                    move to next column;
            end-while
            current column is segmentation column;
        else
            convexity confirmed;
            while (profile moves downward)
                    move to next column;
            end-while
            while (profile moves upward)
                    move to next column;
            end-while
            current column is segmentation column;
        end-if
    end-for
end-algorithm
```

Algorithm 4.4b (Segment _Upper): touching sub-symbols segmentation in upper zone

BEGIN

Step 1: Using the vertical projection in upper zone, identify the boundaries of each sub-symbol. For that whenever $VP(i) = 0$ for $i = 1, 2, 3, \ldots, L$, it is marked as the boundary of sub-symbol. Let us denote the different characters as $C_1, C_2, \ldots, C_n$. Denote first column of the character as $FC_1, FC_2, \ldots, FC_n$ and Last column of the sub-symbol $LC_1, LC_2, \ldots, LC_n$.

Step 2: For $k = 1$ to n performs the following operations (for each character in upper

zone)

Step 2.1: find the top profile of the sub-symbol. For that, for $j = FC_k$ to LC_k perform the following:

Step 2.1.1: mark the row as X, in which first black pixel in j^{th} column is encountered. Now calculate $TP(j) = LR - X + 1$
where TP is top profile and LR is last row of upper zone.

Step 2.2: for $j = FC_k$ to LC_k perform the following

Step 2.2.1: if ($TP\ (j + 1) \geq TP(j)$) go to step 2.2.2(convexity) else goto step 2.2.5(concavity)

Step 2.2.2: while (($TP(j + 1) \geq TP(j)$) and ($j < LC_k$)) increment j by 1 and repeat step 2.2.2

Step 2.2.3: while (($TP(j + 1) \leq TP(j)$) and ($j < LC_k$)) increment j by 1 and repeat step 2.2.3.

Step 2.2.4: if ($j \leq LC_k$) j marks the segmentation column.

Step 2.2.5: while (($TP(j + 1) \leq TP(j)$) and ($j < LC_k$) increment j by 1 and repeat step 2.2.5

Step 2.2.6: while (($TP(j + 1) \geq TP(j)$) and ($j < LC_k$) increment j by 1 and repeat step 2.2.6.

Step 2.2.7: if ($j \leq LC_k$) j marks the segmentation column

END.

The working of this algorithm is illustrated using an example word given in Figure 4.27. Top profile of each sub-symbol in upper zone is identified as shown in Figure 4.27(c). The top profile is scanned from left to right to examine the presence of concavity/convexity. From Figure 4.27(d), one can see that as the column number increases while moving from left to right the number of pixels in top profile also increases. After few columns, number of pixels starts decreasing. This increase and subsequent decrease of the number of pixels represents the convex shape of the character. Now, going further, whenever the downward trend of the number of pixels changes its direction to upwards that marks the segmentation column.

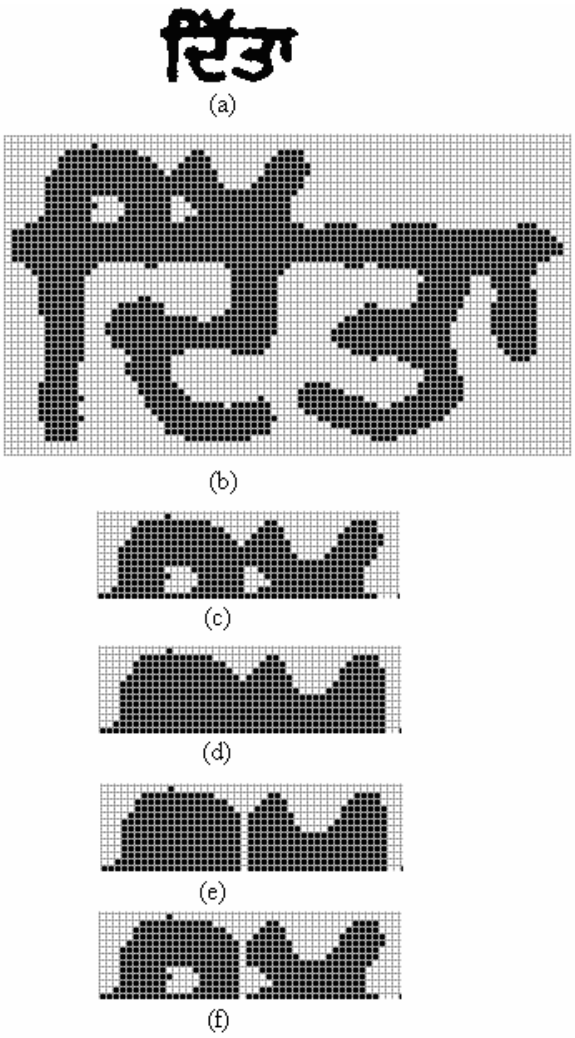


Figure 4.27: Segmentation process: (a) example word, (b) extended view of example word, (c) extended view of problem area, (d) top profile of problem area, (e) segmenting column in top profile, (f) actual segmented sub-symbols

Similar argument holds well when the first character has concavity and in such a situation the number of pixels decreases initially and after few columns it starts increasing. Whenever any downward trend appears it is marked as segmentation column. Second example word in Figure 4.28 shows this concept.

Figure 4.28: Segmentation process in upper zone: (a) example words, (b) problem areas, (c) top profile of problem areas, (d) segmenting columns in top profiles, (e) actual segmented characters

Segmentation problem in this zone becomes difficult when *kanaurā* (◌ੁ) touches with other vowels. As the shape of this character contains one little concavity followed by one little convexity, it produces incorrect segmentation. But the chances of the occurrence of touching sub-symbols involving this sub-symbol are very less. As shown in Figure 4.29, second example word contains *kanaurā* touching with *bindī* and application of Algorithm 4.4 on this touching pair does not produce desired segmentation as shown in Figure 4.29(e). Another kind of problem is that when two touching characters are overlapping and their individual shapes are hard to extract. Under such circumstances, incorrect segmentation occurs as shown in first example word of Figure 4.29.

Figure 4.29: Segmentation problems in upper zone: (a) example words, (b) problem areas, (c) top profile of problem areas, (d) incorrect segmentation for first and third words and no segmentation for second word, (e) segmented sub-symbols after implementing the Algorithm 4.4

Noise may also affect the accuracy of this algorithm. During the process of finding the concavity or convexity, if some noise pixels are present in such a way that it disturbs the concavity or convexity of touching characters, it may also result in incorrect segmentation as shown in third word of Figure 4.29.

We have tried to implement the same algorithm for segmenting touching sub-symbols in printed Devanagari script. Due to structural similarities between Gurmukhi and Devanagari script, Algorithm 4.4 with some modification can be implemented for solving this problem of segmentation of touching sub-symbols in upper zone of Devanagari script also.

In Devanagari script, five symbols (vowels /other strokes) in upper zone have convex shape, *e.g.,* ि ी ै ं ँ and some symbols have convex like shape, e.g., ा ॅ े ै . Only one vowel modifier ॅ contains two little concavities. Although chances of this vowel modifier ॅ touching with other symbols are very less. Algorithm 4.4 works accurately for segmenting touching characters containing convex shaped symbol as first character (shown in Figure 4.30(a) of touching pair. Vowel ॅ contains convex like shape and vowel ॉ contains two small convexities. When Algorithm 4.4 is applied on touching characters involving ॅ , it will produce no segmentation and for vowel ॉ will produce incorrect segmentation. Words containing touching characters involving sub-symbols ॅ and ॉ are shown in Figure 4.30(b). We can modify the Algorithm 4.4 as whenever there is a significant decrease of number of pixels in top profile, and subsequently increase of number of pixels in top profile indicates the segmentation column. Also, when there are three vowels touching in upper zone (shown in Figure 4.30(c)), the algorithm will not segment all the three but only first two characters.

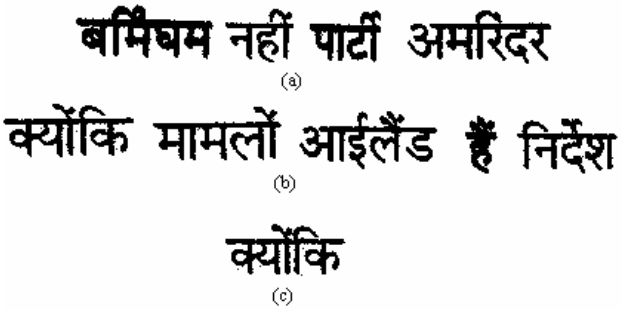


Figure 4.30: Touching sub-symbols in upper zone of Devanagari script: (a) touching sub-symbols which can be segmented using Algorithm 4.4, (b) touching sub-symbols which can be segmented using some modification in Algorithm 4.4, (c) word containing three touching sub-symbols in upper zone

4.9 Segmentation of Touching Sub-symbols in Middle Zone

The majority of touching sub-symbols are found in middle zone of a degraded printed Gurmukhi script document. The above-mentioned (in Section 4.7.2) categories of touching sub-symbols in middle zone are treated individually for segmentation, as discussed below. We have devised Algorithm 4.5 (**Segment _Middle)** to segment touching characters falling in middle zone. This algorithm is based upon the structural properties of Gurmukhi script.

The process of segmenting the touching characters in middle zone proceeds in two passes. In the first pass, for detection of touching position, the structural properties of the Gurmukhi characters have been exploited. Whenever the full sidebar (category b.1) or quarter sidebar (category b.2) is detected it is supposed to be the candidate of segmentation and we have put a segmentation point after the full and quarter sidebars. Similarly, the existence of half sidebar (category b.3) at extreme right end is considered as the candidate of segmentation and we have put a segmentation point after half sidebar. Most of the touching sub-symbols are segmented using these techniques. In the second pass, for segmenting the touching characters of category b.4 and b.5, the conventional method of aspect ratio for finding the candidate of segmentation and minimum pixel density is used for segmenting the touching characters.

Algorithm 4.5a contains outline of touching characters segmentation algorithm in middle zone while Algorithm 4.5b contains the detailed algorithm.

Algorithm 4.5a: Segment _Middle (binary matrix of sub-symbol in middle zone)

begin

get binary matrix of the input document;
find the position of headline;
find the position of baseline;
find height of middle zone;
find continuous vertical projection (*CVP*)of subsymbol;
for (all columns)
if (CVP is greater than 96% of height of middle zone)
full sidebar detected;
mark segmentation column when sidebar columns terminate.

go-to for loop;

end-if

if (CVP is greater than 85% of height of middle zone)

quarter sidebar detected;

mark segmentation column when quarter sidebar columns terminate;

go-to for loop;

end-if

if (CVP is greater than 40% and less than 60% of height of middle zone)

half sidebar detected;

mark segmentation column when half sidebar columns terminate and CVP is less than 20% of height of middle zone;

end-if

end-for

end-algorithm

Algorithm 4.5b (Segment _Middle): touching sub-symbols segmentation in middle zone

BEGIN

Step 1: Recognise the headline. In order to identify the location of headlines, find

$MAXPIX = \max \{HP(i)\}, i = 1, 2, 3, \ldots, L$

The headlines are considered as those lines whose $HP(i) \geq 70\%$ of $MAXPIX$(The threshold limit of 70% is arrived at after detailed and careful experimentation). Let us denote starting location of headlines as SHL_1, SHL_2, $SHL_3, \ldots, SHL_n$ and the ending location of the headlines as EHL_1, EHL_2, EHL_3, …, EHL_n.

Step 2: For $i = 1$ to $LINE_NO$ (where $LINE_NO$ denotes the total number of lines in the input binary document as found in algorithm 1) repeat the following steps:

Step 2.1: recognise individual words by considering $VP(j)$ for $j = 1, 2, 3, \ldots, M$, from $FR(L_i)$ to $LR(L_i)$(first row and last row of i^{th} line denoted as $FR(L_i)$ and $LR(L_i)$. Whenever $VP(j) = 0$, it denotes a word boundary. Denote the individual words as $W_1, W_2, W_3, \ldots, W_p$. First and last column of each word are denoted as $FC(W_j)$ and $LC(W_j), j = 1, 2, 3, \ldots, p$.

Step 2.2: for $k = 1$ to p performs the following operation:

Step 2.2.1: recognise the headline for individual word. For that find $HP(t)$, $\{t = SHL(L_i) - 4$ to $EHL(L_i) + 4\}$, between $FC(W_k)$ to $LC(W_k)$. Find $MAXPIX1 = \max\{HP(t)\}$, $t = SHL(L_i) - 4$ to $EHL(L_i) + 4$. The headlines are considered as those lines whose $HP(t) \geq 90\%$ of $MAXPIX1$. Let us denote starting location of headline for word k as FHW_k and the ending location of the headline for word k as LHW_k.

Step 2.2.2: identify the base line row of the word k by noting the $CVP(j)$, $\{j = FHW_k$ to $LR(L_i)\}$. The location where $CVP(j)$ ends, mark it as α, every time. The row in which maximum α's are found is considered to be the baseline. Mark it as $BASE_k$. Also set height of the middle zone as $HGT_MID = BASE_k - LHW_k$.

Step 2.2.3: note the Continuous vertical projection $CVP(m)$, $m = FC(W_k)$ to $LC(W_k)$, between EHW_k to $BASE_k$

Step 2.2.4: for $g = FC(W_k)$ to $LC(W_k)$ perform the following steps:

Case 1: (category 1)

Step 2.2.4.1: if ((number of pixels in $CVP(g)$) $\geq (0.96 \times HGT_MID)$ (Full sidebar column detected, first category) go to step 2.2.4.2 else go to step 2.2.4.4.

Step 2.2.4.2: while $(CVP(g) \geq 85 \times HGT_MID / 100)$, $g = g + 1$

Step 2.2.4.3: g marks the column where segmentation point to be inserted to segment touching characters of first category. Go to step 2.2.4 //for next sidebar (full, quarter or half) in the same word.

Case 2: (category 2)

Step 2.2.4.4: if $(CVP(g) \geq (0.85) \times HGT_MID)$ (quarter sidebar column detected, Fourth category) go to step 2.2.4.5 else go to step 2.2.4.7

Step 2.2.4.5: while $CVP(g) \geq 75 \times HGT_MID / 100$, $g = g + 1$

Step 2.2.4.6: g marks the column where segmentation point to be inserted to segment touching characters of second

category. Go to step 2.2.4 //for next sidebar in the same word.

Case 3: (category 3)

Step 2.2.4.7: if number of pixels in $CVP(g) \geq (0.40) \times HGT_MID$ and $CVP(g) \leq (0.60) \times HGT_MID$ (half sidebar column detected, third category) go to step 2.2.4.8 else go to step 2.2.4

Step 2.2.4.8: while $CVP\ (g+1) \geq 20 \times HGT_MID\ /\ 100$, $g = g + 1$

Step 2.2.4.9: g marks the column where segmentation point to be inserted to segment touching characters of third category. Go to step 2.2.4 //for next sidebar in the same word.

Step 2.2.5: go to step 2.2 //for next word

Step 3: Go to step 2. // for next line

END

4.9.1 Solution for segmenting touching sub-symbols falling in first category

For segmenting the characters of a word having touching sub-symbols of the first category, we have developed case 1 of Algorithm 4.5. Figures 4.31a-4.31c, explain the working of this algorithm for segmenting touching sub-symbols of first category. Horizontal and vertical projections of a word having touching sub-symbols are given in Figure 4.31a. Furthermore, start of the headline and end of the headline in Figure 4.31b have been marked by white marks in horizontal projection area. The possible locations of sidebar columns in Figure 4.31b are marked by white marks in vertical projection area. We can put a white line after these locations and segmentation is achieved as shown in Figure 4.31c.

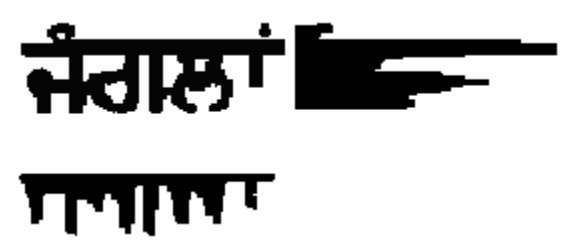


Figure 4.31a: Horizontal and vertical projection of a word containing touching characters

Figure 4.31b: White dots showing start of headline, end of headline and possible locations of sidebar columns

Figure 4.31c: Segmented touching characters using case 1 of algorithm 4.5

This algorithm is based upon the structural properties of Gurmukhi script, that in all the Gurmukhi characters if sidebar exists, it is always present at the extreme right end of the character, in contrary with Devanagari and Bangla script, where it may lie in the middle of the character. The major advantage of this algorithm is that in first pass, we have used the structural properties of the Gurmukhi characters to identify the candidate for segmentation. We have used the aspect ratio principal to find candidate of segmentation in second pass, when most of the touching characters have already been segmented in first pass. Previously in the literature, algorithms used for segmenting touching characters are totally based upon the concept of aspect ratio of the characters. A method of character segmentation was used only if the aspect ratio of a character is greater than some predefined threshold. This idea does not always work well. Furthermore, the methods discussed in literature do not deal with more than two touching characters within a single word. The methods proposed in literature also fail, if size of touching blob is equal or greater than the stroke width. Algorithm 4.5

solves these problems as more than two touching sub-symbols in a single word can be segmented using this algorithm and if the width of touching blob is greater than or equal to the width of the stroke, even then, this algorithm works.

4.9.2 Solution for segmenting touching sub-symbols falling in second category

Case 2 of Algorithm 4.5 has been developed to segment touching sub-symbols falling in this category. The characters falling in second category consists of the sidebar of height more than 85% of the total height of the sub-symbol. Whenever such a column occurs, we continue for looking more consecutive columns. When we get a column whose height is less than 75% of the height of the sub-symbol we put a segmentation mark for this category of touching sub-symbols.

4.9.3 Solution for segmenting touching sub-symbols falling in third category

A challenging task in segmenting touching sub-symbols falling in this category is how to identify the little sidebar, which is approximately half of the height of the middle zone. We have developed case 3 in Algorithm 4.5 to segment touching sub-symbols characters falling in this category. Whenever half sidebar is detected, we put a segmentation mark after the termination of half sidebar.

Case 3 of the algorithm sometimes fails producing over segmentation. The reason behind this is that there are some characters in Gurmukhi script, which have little sidebar at their middle or at the extreme left end. These characters are ਝ, ਠ, ਨ. A solution for this problem has been implemented by considering the fact that whenever we are encountered in case 3, after terminating of half sidebar columns, the transitions from white to black are noted. If two transitions occur then we ignore the half sidebar column (it will be from ਝ, ਠ, ਨ characters) otherwise segment touching sub-symbols.

4.9.4 Solution for segmenting touching sub-symbols falling in fourth category

After implementing the above-mentioned algorithm, we look for candidate of segmentation by considering the aspect ratio of the characters. For that sum of the width of all the characters in document is noted. Average width of a sub-symbol (*AWC*) is found by dividing the total width by number of sub-symbols. Any sub-symbol whose width is more than 150% of *AWC* is considered to be the merged sub-symbol and candidate of segmentation. Now, for segmenting touching sub-symbols of these candidates, we look for the density of the pixels in columns from left one third to right one-third columns of the candidate sub-symbols. Wherever the density of pixels is minimum we consider it as segmentation column. Figure 4.32 contains some words containing touching sub-symbols falling in fourth category and the problem areas have been encircled.

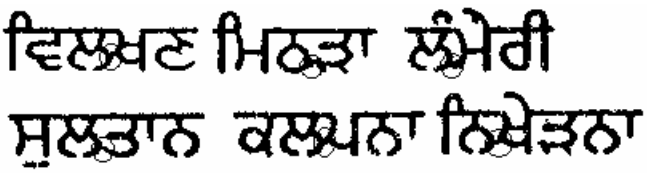


Figure 4.32: Touching sub-symbols falling in fourth category (problem area encircled)

After implementing the above mentioned algorithm, one is able to segment about 76-91% of the total touching sub-symbols. Over segmentation occurs in few cases and few times the algorithm is unable to segment touching pair and bypasses it without segmenting. The major drawbacks and problems, we faced during segmentation using this algorithm are shown in Figure 4.33 and explained below.

Sometimes, a character has a stroke similar in shape of half, full or quarter sidebar, as shown in Figure 4.33(a) in middle or left side of the character. Since Algorithm 4.5 (case 1, 2 and 3) is based on the concept of sidebar, it results in over segmentation, by considering a non sidebar stroke as sidebar stroke. Identifying the candidate of segmentation is not possible in some cases as shown in Figure 4.33(b) and 4.33(c). This is due to the fact that the width of touching pair of sub-symbols is comparable to the two widest sub-symbol in Gurmukhi script (ਘ, ਅ). A solution to this problem has been found as both of these sub-symbols do not contain any headline. So this concept can be used to identify that weather a wide sub-symbol

is actually a touching pair (contains headline) or a single sub-symbols (ਘ, ਅ) containing no headline.

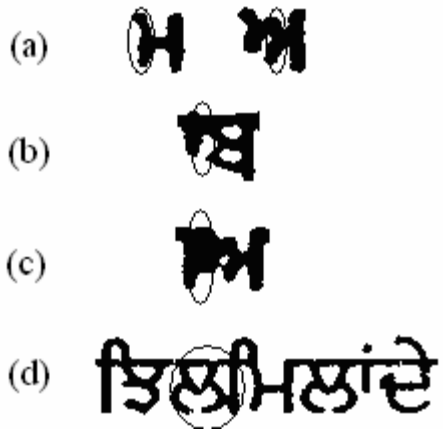


Figure 4.33: Problems in segmenting sub-symbols in middle zone

Similarly, as shown in Figure 4.33(c), even though it is identified that this character is a touching pair using its aspect ratio, but touching blob is very much big and it may result in wrong segmentation.

4.10 Segmentation of Touching Sub-symbols in Lower Zone

As described in Section 4.7.3, touching sub-symbols in lower zone can be divided into three categories. For category c.1, it is sufficient to identify the base line of a strip. Base line segments the middle zone sub-symbols from lower zone sub-symbols. As shown in Figures 4.34(a-d), middle zone sub-symbols have been segmented from lower zone sub-symbols by identifying the base line. For category c.2, candidate of segmentation can be identified by the aspect ratio of the sub-symbols. The character having aspect ratio greater than a threshold value is considered to be candidate of segmentation. For actual segmentation, one can use the same technique as used for segmenting touching sub-symbols of category b.4 in Section 4.9.4. Identification of the pattern of category c.3 is sufficient and needs no segmentation.

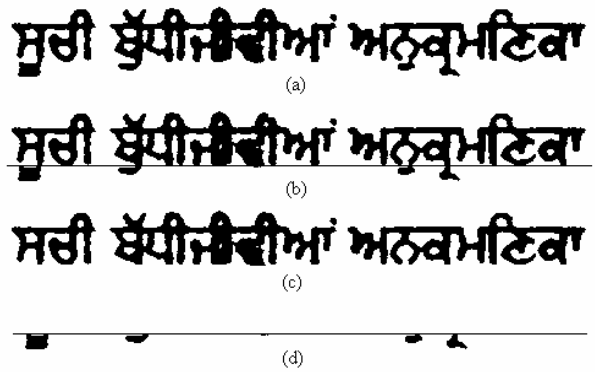


Figure 4.34: Segmentation of touching sub-symbols of category c.1 in lower zone: (a) line containing touching sub-symbols of category c.1, (b) base line identified, (c) middle zone sub-symbols separated from lower zone, (d) lower zone sub-symbols separated from middle zone sub-symbols

4.11 Results and Discussion

For implementing the algorithms proposed in this chapter, we selected degraded documents containing touching characters from various books and magazines as well as normal documents, faxed them, xeroxed them and scanned them at 300 dpi resolutions. About 250 such documents were scanned which contains almost 11000 touching characters, thus a sufficiently large database of touching characters has been created. Figure 4.35 shows one paragraph taken from this database. This paragraph contains touching characters in middle, upper and lower zones.

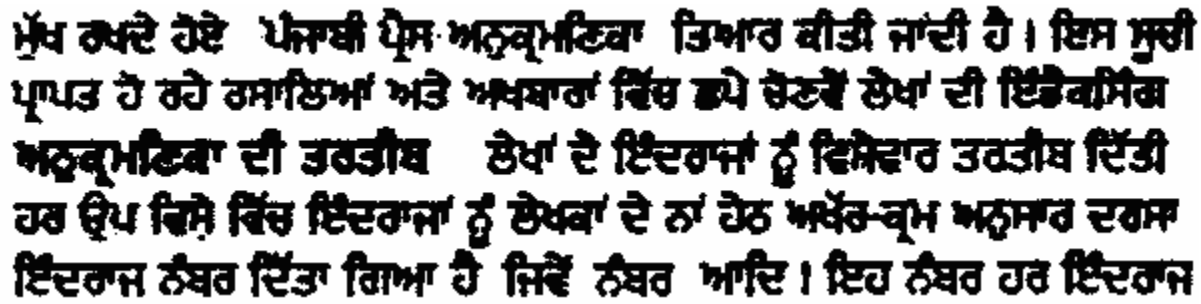


Figure 4.35: Gurmukhi paragraph containing touching characters and heavily printed characters

The main objective of this chapter was to discuss a complete solution for segmentation phase including line, word and character segmentation of degraded printed Gurmukhi script

documents. As the problem of word segmentation is trivial, we have discussed in detail character segmentation and line segmentation problems which include methods to segment the multiple horizontally overlapping lines and segment touching characters present in all the three zones in degraded printed Gurmukhi script.

For segmenting horizontally overlapping lines, we have scanned a number of documents taken from leading newspapers of Gurmukhi script. The problem of overlapping lines and existence of broken components of a line (small strips) is found even in good quality newspapers. Algorithm 4.1 proposed in this chapter is useful for segmenting multiple horizontally overlapping lines in printed Indian scripts. This algorithm also joins together the broken components of an over segmented line. Various types of strips and the percentage of occurrence of these strips have been calculated for eight major Indian scripts as illustrated in this chapter. The entire database has been prepared scanning single column news articles from eight printed Indian script newspapers. Algorithm 4.2 (**Segment_Lines_Scripts)** segments the overlapping lines accurately 95%-99.7% times for various scripts. As given in Table 4.2, the problem of more than two overlapping lines exists in Gurmukhi script documents and the available methods do not solve this problem. Algorithm 4.2 proposed by us tackles this problem very efficiently. This algorithm is also tested on documents taken from poorly printed magazines and old documents of Gurmukhi script. The same algorithm has also been used to segment the horizontally overlapping lines and associating broken components of a line to that line for seven other Indian scripts, namely, Devanagari, Bangla, Gujarati, Kannada, Tamil, Telugu and Malayalam. The overlapping lines in the different sized text in printed newspapers in Gurmukhi script have correctly been segmented 98.12% of the times. The specific problem in this case has been identified as the situation when there are two consecutive lines of different font size. This problem has successfully been solved in the proposed Algorithm 4.3. The algorithms discussed in literature to segment overlapping lines fail to solve the problem of segmenting the overlapping lines of different size text [70, 71].

We have also solved the challenging problem of segmentation of merged characters in printed degraded Gurmukhi text. For this purpose, the merged characters in each zone are segmented individually. Separate algorithms (4.4, 4.5) have been developed for all the three zones. Algorithms have been tested on 56 degraded printed Gurmukhi script documents. For

segmenting touching characters in middle zone, we have implemented Algorithm 4.5. The results on eight representative degraded printed Gurmukhi script documents are given in Table 4.11. The percentage accuracy of segmentation for these documents is in the range of 77-91%.

Table 4.11: Percentage accuracy in middle zone

Document	Total sub-symbols	Number of touching sub-symbols in category				Number of Correctly segmented sub-symbols	Missed Segmentation (percentage)	Over-segmentation (percentage)	Percentage Accuracy
		b.1	b.2	b.3	b.4				
Doc1	955	39	12	14	2	56	5.97	10.45	83.58
Doc2	1023	46	9	30	13	81	5.10	12.25	82.65
Doc3	984	105	26	21	18	131	7.06	15.88	77.06
Doc4	1023	119	33	41	25	200	3.21	5.05	91.74
Doc5	1045	125	34	36	27	189	4.50	10.36	85.14
Doc6	992	57	7	14	22	86	5.00	9.00	86
Doc7	923	41	6	17	6	58	2.86	14.28	82.86
Doc8	776	14	4	2	15	32	0	8.57	91.43

Similarly, for segmenting touching characters in upper zone we have implemented Algorithm 4.4 proposed in Section 4.8 and observed that an accuracy of 76-92% has been achieved. The results are shown in Table 4.12.

Table 4.12: Percentage accuracy in upper zone

Document	Total characters in upper zone	Number of Touching sub-symbols	Number of Correctly segmented Sub-symbols	Missed Segmentation (percentage)	Over-segmentation (percentage)	Percentage accuracy
Doc1	307	19	15	5.26	15.79	78.95
Doc2	357	15	12	0	20	80.0
Doc3	319	22	20	4.55	4.55	90.90
Doc4	297	17	13	5.88	17.65	76.47
Doc5	324	20	16	10	10	80.00
Doc6	267	14	12	0	14.29	85.71
Doc7	243	18	15	5.56	11.11	83.33
Doc8	189	10	8	10	10	80.00

For segmenting touching characters of first category in lower zone, the technique given in Section 4.10 has been implemented. It has been observed that 92% of lower zone characters

are correctly segmented from middle zone characters by using the concept of base line. We have also achieved an accuracy of 96% for segmenting touching sub-symbols of category c.2 in lower zone.

Figure 4.36a contains part of Gurmukhi script document containing touching characters and heavily printed characters. Figure 4.36b contains results of standard character segmentation algorithms applied on the document. Each sub-symbol has been shown by different color. One can see in Figure 4.36b that most of touching sub-symbols have not been segmented correctly. We have encircled few non-segmented sub-symbols.

Figure 4.36c contains the output of the document after the application of Algorithm 4.5. One can see that most of touching sub-symbols have been correctly segmented. It can also be noted that, one sub-symbol has been over segmented. The over segmented character has been encircled.

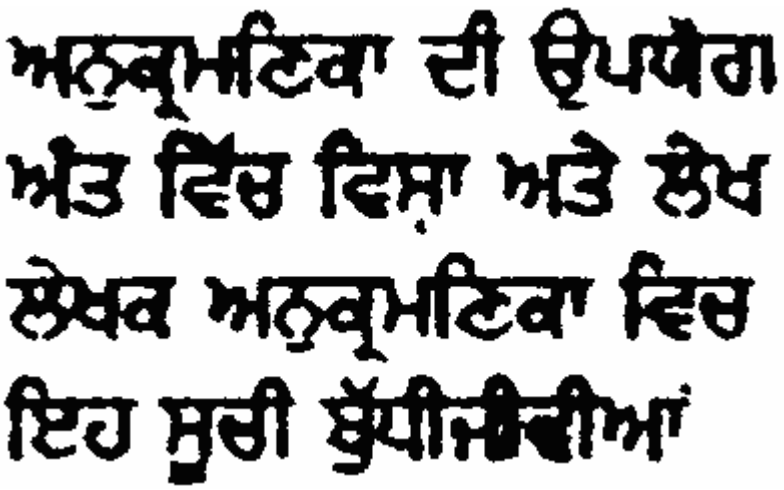


Figure 4.36a: Part of Gurmukhi script document

Figure 4.36b: Segmented characters using standard character segmentation algorithms producing incorrect segmentation

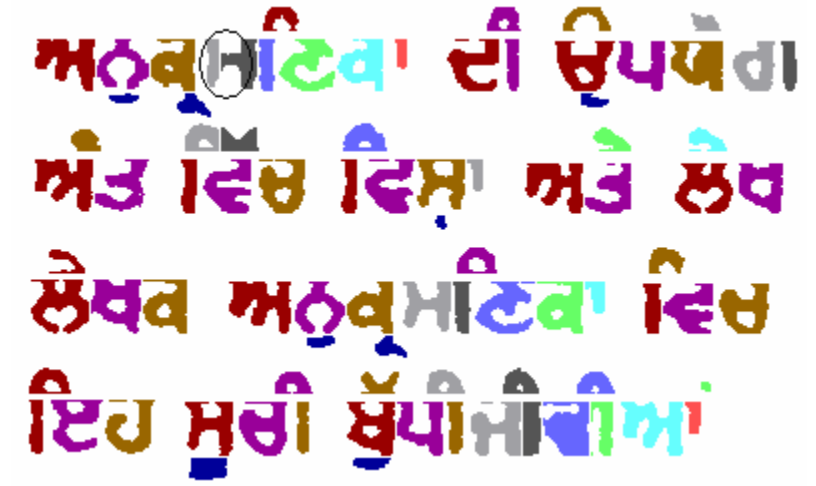


Figure 4.36c: Segmented characters using proposed algorithms

Chapter 5
Feature Extraction

Feature extraction is the first step of an OCR engine as described in Chapter 1. The performance of a character recognition system depends heavily on what type of features and how many features are being used. In machine-printed documents, shape discrepancy among characters belonging to the same prototype is sometimes quite large because of the poor quality and low resolution of the document images. Particularly, when touching characters are segmented, the noise blobs near the cutting points overlap both sides of the characters, possibly resulting in a large dissimilarity between the input pattern and the corresponding sample class. These noise blobs become the part of the character and it is impossible to remove touching noise blobs by simple techniques.

Therefore, it is required to select features which can adapt the shape variations due to touching noise blobs. In fact, the main problem in OCR system is the large variation in shapes within a class of character. This variation depends on font styles, document noise, photometric effect, document skew and poor image quality. The large variation in shapes makes it difficult to determine the number of features that are convenient prior to model building. Though many kinds of features have been developed and their test performances on standard database have been reported, there is still room to improve the recognition rate by developing an improved feature.

A feature extraction method should be invariant to certain transformations such as translation, scaling, rotation, stretching, skewing and mirroring [79]. Invariance to contrast is also required for gray scale images. Reconstructability is another requirement of good feature extraction method. The extracted features are considered to be valuable if characters can be reconstructed from the extracted features.

Features can be grouped into three classes depending on whether they are extracted from the whole word (high level features), the characters (medium level features) or sub-characters

(low level features). Low level features are extracted from letter fragments that have elementary shapes such as small lines, curved strokes, bars *etc.* Features account, in general, for their position and simple geometric characteristics.

Features such as loops, ascenders and descenders are often referred as high level features. Since they consist of the detection of structural elements, they do not depend on the writing style and are then stable with respect to cursive variability. Together with loops, ascenders and descenders (the most used since they are easily detected), we also find junctions, stroke endpoints, *t*-bars and dots in the literature.

As discussed in Chapter 2, there is substantial number of feature extraction techniques detailed in the literature for character recognition. They may be loosely categorized into global and local (topological) feature extraction methods. As already discussed, the global methods are very simple to implement, and model the global characteristics of a character. One of its main advantages is that it ignores local noise or distortions in the character image. Conversely, topological feature extraction techniques examine the geometry and topology of the character, *e.g.*, stroke direction, convexities, junction points *etc.* In fact, topological feature extraction techniques have proven to be the most popular amongst researchers for handwritten character recognition. The sections below describe the two feature extraction techniques that were investigated in this research. The first is based on the extraction of topological features whereas the second is a global technique.

5.1 Structural Features

Structural features may be defined in terms of character strokes, character holes, or other character attributes such as concavities and convexities, end points and junctions, extrema, intersection with straight lines *etc.* Structural features can be classified into two categories.

1. Local features which are usually geometric. (*e.g.*, concave/convex parts, number of endpoints, branches, joints *etc.*)
2. Global features, which are usually topological (connectivity, projection profile, number of holes, *etc.*)

   Structural features have the following advantages.

1. It is intuitive, meaning that the designer of a structure method has full control over the parameters and the fine details of the process. In contrast, the results and the performance of a statistical method, depend heavily on the parameters, features set used, and the training set.
2. It can compensate for heavy variations in the input. The structural approach possesses the capability to deal with heavily distorted data.
3. Structural methods can be designed to take advantage of the whole shape definition of an input character. The statistical approaches look only at predefined feature vectors, which provide only partial information about the shape.

Structural features should be chosen keeping in mind that the shape variations should affect feature set minimally. It was not an easy task to decide which structural features should be chosen to extract the structural features from degraded characters of Gurmukhi script due to large shape variations in characters of the same class. Feature codes of the structural features set have following common characteristics.

- These structural features are less sensitive to character size and font.
- The feature codes present a very high separability for different characters. In other words, the feature codes representing different characters have a very low probability to coincide.
- These features are very much tolerant to noise.
- Features are less sensitive to character variations, due to font differences or scanning effects.

We have used the following structural features of Gurmukhi characters for constructing feature vector.

1. Presence of sidebar (St1): This feature is present if a vertical sidebar, of approximately the same height as of the character, is present at the rightmost side of the sub-symbol. As discussed in Chapter 4, we have already used this feature for segmentation purpose. Further,

it is noted that if full sidebar exists in Gurmukhi characters, it is always at the rightmost side of the character. There are 11 characters in middle zone having full sidebar at their right end. These characters are: ਅ, ਸ, ਖ, ਘ, ਜ, ਥ, ਧ, ਪ, ਬ, ਮ, ਯ. Additionally, second component 'ਾ' of the multicomponent character *gaggā* (ਗ) has sidebar at the right end. There are two vowels *bihārī* (ੀ) and *sihārī* (ਿ), whose one stroke falls in middle zone having sidebar at their right end. Furthermore, as discussed in Chapter 4, four characters have quarter sidebar at their right end. These characters are: ਰ, ਹ, ਚ and first component 'ਰ' of the multicomponent character ਗ. These characters have also been considered for this feature. As such, there are total 18 sub-symbols (characters, vowels, components of multicomponent characters) containing this feature. This feature divides the whole set of sub-symbols in middle zone in almost two equal sized subsets. The feature is true if a vertical sidebar is present at rightmost side of the sub-symbol else it is false. As shown in Figure 5.1(a), the Gurmukhi character *sassā* (ਸ) has full sidebar.

2. Presence of half sidebar (St2): This feature is present if a sidebar, of approximately half the height of the full character is present at the rightmost side of the sub-symbol. As discussed in Chapter 4, we have already used this feature for segmentation purpose. There are 8 characters in Gurmukhi script having this feature: ੲ, ਛ, ਢ, ਟ, ਫ, ਦ, ਫ, ਵ. In addition to these, one vowel *kannā* (ਾ) also contains this feature. This feature is true if a half sidebar is present at the rightmost side of the sub-symbol else it is false. As shown in Figure 5.1(b), for Gurmukhi character *īrī* (ੲ) this feature is true.

3. Presence of headline (St3): The presence of headline in the sub-symbol is another important feature for classification. Even when the sub-symbols are highly degraded, this feature is retained. For example, ਪ has no headline while ਟ has headline. There are 30 sub-symbols in middle zone of Gurmukhi character set having this feature present: ੳ, ੲ, ਸ, ਹ, ਕ, ਗ, ਙ, ਚ, ਛ, ਜ, ਝ, ਞ, ਟ, ਠ, ਡ, ਢ, ਣ, ਤ, ਥ, ਦ, ਧ, ਨ, ਫ, ਬ, ਭ, ਯ, ਰ, ਲ, ਵ, ੜ. Furthermore, three vowels ੀ, ਿ and ਾ have this feature true. This feature is very much robust to the noise. This feature is extremely useful for differentiating similar characters such as ਸ and ਮ, ਖ and ਥ, ਧ and ਪ. This feature is true if headline is present else it is false. As shown in Figure 5.1(c), the Gurmukhi character ਸ contains this feature.

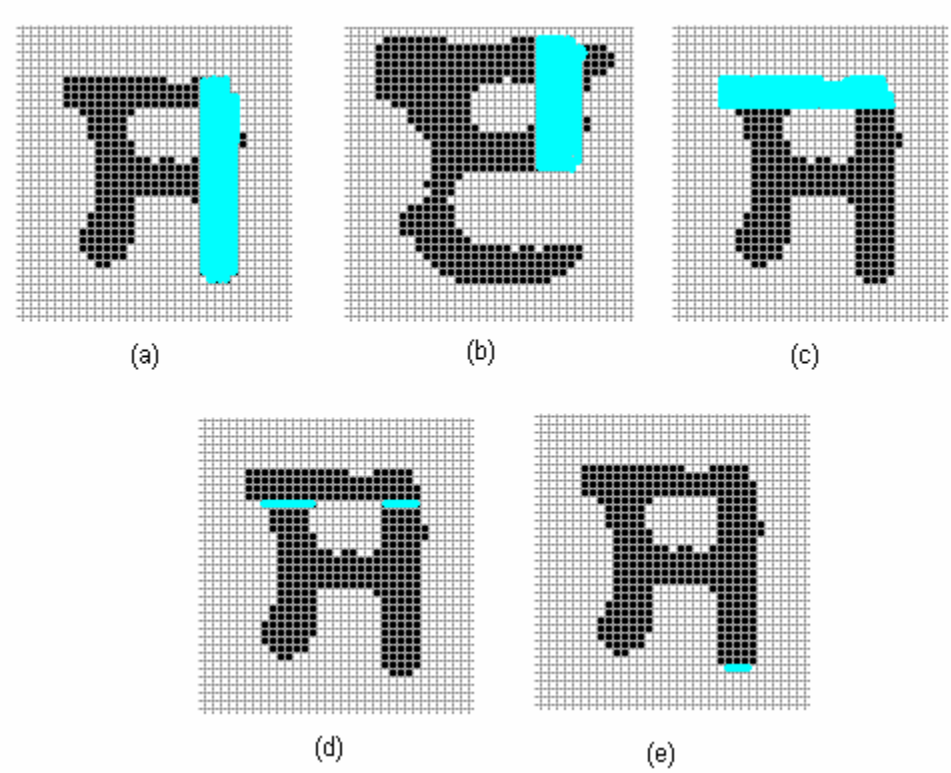


Figure 5.1: Structural features of Gurmukhi characters: (a) sidebar present, (b) half sidebar present, (c) headline present, (d) two junctions with headline, (e) one junction with baseline

4. Number of junctions with headline (St4): It can be noted that each character in middle zone of Gurmukhi character set has either one or more than one junctions with the headline. For example ਰ has one junction with headline while ਯ has two junctions with headline. This feature is true if a sub-symbol of Gurmukhi has one junction with headline else it is false. There are 19 sub-symbols in Gurmukhi characters set having this feature true: ਹ, ਕ, ਙ, ਚ, ਛ, ਜ, ਟ, ਠ, ਡ, ਢ, ਣ, ਤ, ਦ, ਨ, ਫ, ਭ, ਰ, ਵ, ੜ. Additionally, this feature is true for both the components of ਗ (a multicomponent character). Moreover, this feature is also true for vowels ੀ, ਿ. On the contrary, this feature is false for ੳ, ਅ, ਏ, ਸ, ਖ, ਘ, ਝ, ਵ, ਥ, ਧ, ਪ, ਬ, ਮ, ਯ and ਲ. This feature has been used to divide the complete Gurmukhi characters set into two almost equal sized subsets. As shown in Figure 5.1(d), for the Gurmukhi character ਸ, this feature is false. Unfortunately, heavy degradation of a sub-symbol can wrongly calculate this feature. As shown in Figure 5.2, for the characters ਕ, ਚ, ਡ, ਢ, ਦ and ਵ, this feature has become false instead of true.

Figure 5.2: Degraded Gurmukhi characters having feature St4 false instead of true

5. Number of junctions with the baseline (St5): This feature is true for a sub-symbol if number of junctions with the base line is one else it is false. This feature is true for sub-symbol ਰ and false for sub-symbol ਜ since it has two junctions with the baseline. Further, only single vowel in middle zone named, *kannā* (ਾ), do not have any junction with the baseline. Therefore, for sub-symbol *kannā*, this feature is also false. There are 26 sub-symbols in Gurmukhi characters set having this feature true: ੳ, ੲ, ਹ, ਸ, ਕ, ਙ, ਚ, ਛ, ਝ, ਵ, ਟ, ਠ, ਡ, ਢ, ਣ, ਤ, ਥ, ਦ, ਧ, ਪ, ਫ, ਬ, ਭ, ਰ, ਵ, ੜ. This feature is false for ਅ, ਘ, ਜ, ਨ, ਮ, ਯ and ਲ characters. Most of the times, sub-symbols ਅ, ਸ and ਘ show variability for this feature as sometime this feature is true and sometimes false for these sub-symbols. As shown in Figure 5.1(e), for Gurmukhi character ਸ, the feature St5 is true.

6. Aspect ratio (St6): Aspect ratio is obtained by dividing the height of the character by the width of the character. We have divided the whole sub-symbols present in middle zone into three categories depending upon the aspect ratio of the sub-symbols. We consider St6 = 0 if the aspect ratio is less than 0.90. There are two wider characters ਅ and ਘ in Gurmukhi script having St6 = 0. Also if aspect ratio is greater than 3.0 then St6 = 2. There are three vowels ੀ, ਿ, ਾ and second component of the multicomponent character ਗ having very high aspect ratio (>3.0). For all other sub-symbols in middle zone the value of St6 is considered as 1. Also, the value of aspect ratio has been used in case of upper zone and lower zone sub-symbols.

7. Left, right, top and bottom profile direction codes (St7, St8, St9 and St10): A variation of chain encoding is used on left, right, top and bottom profiles. For finding the left profile direction codes, the left profile of a sub-symbol is scanned from top to bottom and local directions of the profile at each pixel are noted. Starting from current pixel, the pixel distance of the next pixel in east, south or west directions is noted. The cumulative count of movement in three directions is represented by the percentage occurrences with respect to the

total number of pixel movement and stored as a 3 component vector with the three components representing the distance covered in east, south and west directions, respectively. The direction code of the profile of Figure 5.3 is {30, 20, 50}, since the movements in east, south and west directions are 3, 2 and 5 pixels, respectively. Table 5.1 illustrates the distance covered in pixels as one moves from row number 1 to row number 10. Similarly, right profile direction codes are found by scanning right profile from top to bottom and movement is noted in east, south and west directions. Furthermore, for finding the direction code of the top and bottom profiles, east, south and north directions are considered while moving from left to right. As such, a total of twelve (4×3) structural features are obtained using this feature. This feature gives the movements of the strokes along the external boundaries of the sub-symbols.

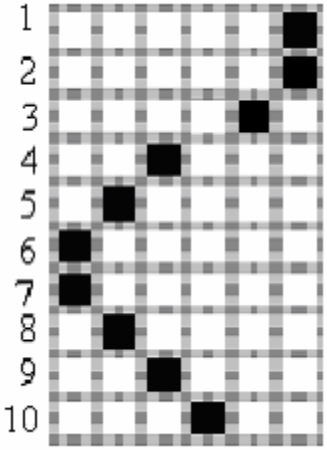


Figure 5.3: Projection profile of a sub-symbol

Table 5.1: Calculation of feature St7

Movement from row number to row number	Distance covered in pixels		
	East	South	West
1 – 2		1	
2 – 3			1
3 – 4			2
4 – 5			1
5 – 6			1
6 – 7		1	
7 – 8	1		
8 – 9	1		
9 – 10	1		
Total	**3**	**2**	**5**

This feature is very useful in the case of degraded text recognition. As shown in Figure 5.4(a), the sub-symbol *kakkā* **(**ਕ) has a loop in its structure and as shown in Figure 5.4(b), the loop has been filled due to degradation. In both the cases, all left, right, top and bottom profile direction codes will produce the same feature values.

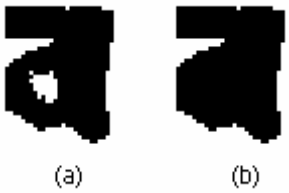


Figure 5.4: Gurmukhi character *kakkā* (ਕ): (a) degraded character having loop, (b) degraded character having loop filled

8. Directional Distance Distribution (St11): Directional Distance Distribution (DDD) is a distance based feature proposed by Oh and Suen [167]. For every pixel in the input binary array, two sets of 8 bytes which are called W(White) set and B(Black) set are allocated as shown in Figure 5.5. For a white pixel, the set W is used to encode the distances to the nearest black pixels in 8 directions (0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, 315^0). The set B is simply filled with value zero. Similarly, for a black pixel, the set B is used to encode the distances to the nearest white pixels in 8 directions. The set W is filled with zeros. In Figure 5.5, the color of pixel at coordinates (6, 6) is white. For the direction 0^0, the traveled sequence is: (6, 6)W $\rightarrow$ (6, 7)W $\rightarrow$ (6, 8)W$\rightarrow$ (6, 9)B. The traveled distance 3 is recorded for 0^0. Sometimes, we are encountered with boundary of the array without finding the black pixel. At this stage, array is supposed to be circular. Therefore, while finding the nearest black pixel in 225^0 direction, the following travel sequence will be followed: (6, 6)W $\rightarrow$ (7, 5)W $\rightarrow$ (8, 4)W $\rightarrow$ (9, 3)W $\rightarrow$ (10, 2)W $\rightarrow$ (11, 1)W $\rightarrow$ (1, 11)W $\rightarrow$ (2, 10)W $\rightarrow$ (3, 9)W $\rightarrow$(4, 8)W $\rightarrow$ (5, 7)B, and the traveled distance 10 is recorded. The set B is simply filled with zeros as shown in Figure 5.6(a). Similarly, for the black pixel at coordinates (4, 6), the directional distance values have been shown in Figure 5.6(b). The distances of nearest black/white pixel in each direction for pixels (6, 6) and (4, 6) have been given in Table 5.2. After computing WB encoding for each of the pixel, we have divided the input array into four equal zones both horizontally and vertically, hence producing 16 zones. We have taken

the average of WB encoding in each of these 16 zones. Finally, we got a 16 × 16 feature vector.

We have used the non-linear normalization method proposed by Lee and Park [168] to normalize the sub-symbols. Each sub-symbol has been normalized to a 50 × 50 matrix.

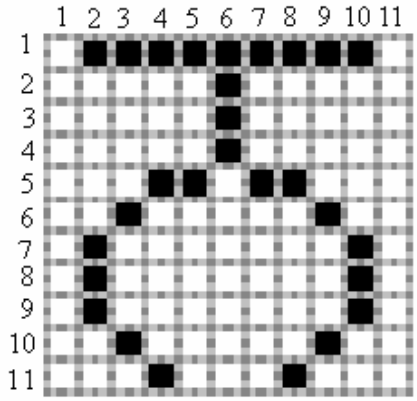


Figure 5.5: Projection profile of a Gurmukhi character

w0	w1	w2	w3	w4	w5	w6	w7	b0	b1	b2	b3	b4	b5	b6	b7
3	1	2	1	3	10	6	11	0	0	0	0	0	0	0	0

(a)

w0	w1	w2	w3	w4	w5	w6	w7	b0	b1	b2	b3	b4	b5	b6	b7
0	0	0	0	0	0	0	0	1	1	4	1	1	2	1	2

(b)

Figure 5.6: Example of WB encoding: (a) WB encoding for the white pixel at (6, 6), (b) WB encoding for the black pixel at (4, 6)

Table 5.2: Calculation of feature St11

Directions (in degree)	**Distance of nearest black pixel for pixel (6,6)**	**Distance of nearest white pixel for pixel (4,6)**
0	3	1
45	1	1
90	2	4
135	1	1
180	3	1
225	10	2
270	6	1
315	10	2

9. Transition features (St12): In this structural feature, location and number of transitions from background to foreground pixels in the vertical and horizontal directions are noted. The transition feature used here is similar to that proposed by Gader *et al.* [169]. To calculate transition information, image is scanned from left-to-right, right-to-left, top-to-bottom and bottom-to-top. To ensure a uniform feature vector size, the transitions in each direction are computed as a fraction of the distance traversed across the image. For example, if the transitions were being computed from top-to-bottom, a transition found close to the top would be assigned a high value compared to a transition computed further down. A maximum value (M) was defined to be the maximum number of transitions that may be recorded in each direction. Conversely, if there were less than M transitions recorded (n for example), then the remaining $M - n$ transitions would be assigned values of 0 (to aid in the formation of uniform vectors).

It will produce four matrices, two matrices having dimensions $NC \times 5$ and other two matrices having dimensions $NR \times 5$ (where NC represents the Number of Columns/width of the character matrix and NR represents the number of Rows/height of the character, and we have considered $M = 5$). The second stage of transition feature calculation consists of resampling the transition locations onto fixed size grids. For that, we have divided each matrix horizontally into T equal parts. We have taken the average of transitions vertically in each part. Finally, if $NC = 50$, $NR = 50$, $M = 5$ and $T = 5$, we got a $4 \times 5 \times 5$ feature vector.

5.1.1 Additional assumptions

The following additional assumptions can be used while detection of the structural features from St1 to St6.

1. If St1 is true for any sub-symbol, then St2 is false, *i.e.*, if full sidebar (St1) is detected, then half sidebar (St2) is absent.
2. If St1 is false, then St6 is not equal to 1, *i.e.*, if full sidebar is not detected, then it can not be wide character having low aspect ratio (for which St6 = 1).
3. If St3 is false, then St4 is also false, *i.e.*, if character has no headline, then the number of junctions with headline is not one.

4. If St6 = 3 and St5 is false, then the character is ਾ as only this single vowel has high aspect ratio (St6=3) and number of junctions with the baseline is not one.
5. If St6 = 3, then St7 = St8 = St9 = St10 = {0, 0, 0}.
6. If St1 is true, then St7 = {0, 0, 0}, *i.e.*, if full sidebar is present then left profile chain code is 0.
7. If headline is absent for a sub-symbol, only then aspect ratio = 1, *i.e.*, if St3 is false and aspect ratio is less than 0.90 only then St6 = 1. As aspect ratio is less than 0.90 (St6 = 1) for only two characters ਅ and ਘ. However, most of the times, sub-symbol ਯ or a touching pair of sub-symbols has aspect ratio < 0.90 and the feature St6 is evaluated wrongly for such sub-symbols. Generally, ਯ or a touching pair of sub-symbols having aspect ratio < 0.90 would have headline. However, as discussed if St6 = 1, headline must be absent. As a result of this, the wrongly calculated value of St6 can be corrected.

5.2 Statistical Features

Statistical features have been also used to extract features from segmented degraded characters. We have used the following statistical features.

5.2.1 Zoning

The extracted character image (raw or re-scaled), is segmented into windows of equal size. Density values (Number_of_foreground_pixels / Total_number_of_pixels) are obtained for each window as shown in Figure 5.7. All density values are used to form the input feature vector for a particular character pattern.

As defined by Trier *et al.* [79], zoning can be defined as the process in which an $n \times m$ grid is superimposed on the character image and for each of the $n \times m$ zones; the average grey levels in case of grey level character images are used as features. In case of binary images the percentage of black pixels in each zone is computed [79]. As zoning is not invariant to scaling, we have scaled the characters before finding features using zoning.

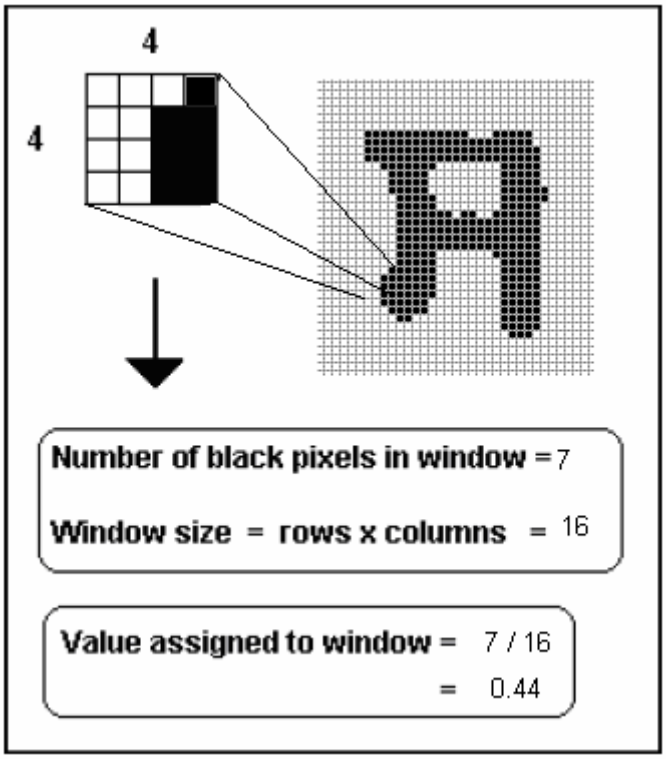


Figure 5.7: Determination of pixel density values for a particular window of the character matrix

5.2.2 Moments

Moments are pure statistical measure of pixel distribution around center of gravity of characters and allow capturing global character shapes information. They describe numerical quantities at some distance from a reference point or axis. They are designed to capture both global and geometric information about the image. Moment-based invariants explore information across an entire image rather than providing information just at single boundary point, they can capture some of the global properties missing from the pure boundary-based representations like the overall image orientation. In 1962, Hu [84] employed the theory of algebraic invariants and derived his seven famous invariants to rotation of 2-D objects. Since that time, moment invariants have become a classical tool for feature-based object recognition. The original Hu's invariants utilized the second and third-order moments only. The construction of the invariants from higher-order moments is not straightforward. Fourier-Mellin transform, Zernike polynomials and algebraic invariants have been used in a number

of applications as given in details by Teh and Chin [86] to achieve invariant recognition of two-dimensional image patterns.

Geometric moments or regular moments are simple to implement. Hu's invariants have the desirable properties of being invariant under image translation, scaling, and rotation. However, it is found that to compute the higher order of Hu's moment invariants is quite complex, and to reconstruct the image from Hu's invariants is also very difficult. A pioneer work on this field was done independently by Reiss [81] in 1991 and by Flusser and Suk [83] in 1994. They corrected some mistakes in Hu's theory, introduced affine moment invariants (AMI's), and proved their applicability in simple recognition tasks. Hu [84] and Flusser and Suk [83] have shown that a set of eleven invariant functions have been widely used. Zernike first proposed the Zernike polynomials in 1934. Their moment formulation appears to be one of the most popular, outperforming the alternatives (in terms of noise resilience, information redundancy and reconstruction capability). Complex Zernike moments have been extensively used as the invariant global features for image recognition. Zernike moments have been analyzed and implemented by Teh and Chin [86] and Khotanzad and Hong [170]. Singh [171] has used floating point calculations for Zernike moments.

We, in the presented work, have used Zernike moments and Orthogonal Fourier Mellin moments as features.

5.2.2.1 Zernike Moments

Zernike introduced a set of complex polynomials $\{V_{nm}(x, y)\}$ which form a complete orthogonal set over a unit disk of $x^2 + y^2 \leq 1$. The form of the polynomial is:

$$V_{nm}(x, y) = R_{nm}(x, y)e^{jm\theta} \tag{5.1}$$

where $j = \sqrt{-1}$, $\theta = \tan^{-1}\left(\frac{y}{x}\right)$

and

$$R_{nm}(x, y) = \sum_{s=0}^{n-|m|/2} \frac{(-1)^s (x^2 + y^2)^{(n-2s)/2} (n-s)!}{s!((n+|m|-2s)/2)!((n-|m|-2s)/2)!} \tag{5.2}$$

where $n \geq 0$, $(n-|m|) = even$, and $|m| \leq n$.

For a digital image the Zernike moments of order *n* and repetition *m* are expressed as:

$$A_{nm} = \frac{n+1}{\pi} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) V_{nm}^{*}(x,y), \tag{5.3}$$

where $x^2 + y^2 \leq 1$ and * denotes the complex conjugate operator.

The defined features of Zernike moments are only invariant to rotation. To achieve scale and translation invariance, the image needs to be normalized by using regular Zernike moments.

The translation invariance is achieved by translating the original image

$f(x, y)$ to $f(x + \bar{x}, y + \bar{y})$, where $\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{01}}{m_{00}}$

The original image's center is moved to the centroid before the Zernike moments calculation.

Scale invariance is achieved by enlarging or reducing or scaling the image about its centroid and then performing the scaling operation to convert the image to a standard size. One of the most important properties of the Zernike moments is its power to reconstruct the image from the calculated moments. Suppose we know all Zernike moments A_{nm} of $f(x, y)$ up to order *N*. Due to the orthogonal property of Zernike moments, we can reconstruct the image based on this set of Zernike moments by:

$$\hat{f}(x, y) = \sum_{n=0}^{n_{\max}} \sum_{m=-n}^{n} A_{nm} V_{nm}(x, y) \tag{5.4}$$

where $n_{\max}$ is the maximum order of the Zernike moments considered.

5.2.2.2 Orthogonal Fourier Mellin Moments

OFMMs, introduced by Sheng and Shen [172], contain more local information about character recognition, which are based on a set of radial polynomials. They can be used to represent small characters in the same way as large characters. As described by Kan and Srinath [173], the number of OFMMs required to represent an image is much lower than that of ZMs so that OFMMs can be more robust than ZMs if the characters have large variability and describe small image more accurately when large size character samples are taken for training and relatively smaller size characters are used for testing.

As Shen and Sheng introduced OFMMs, the circular Fourier or Radial Mellin Moments (FMMs) of an image function $f(r, \theta)$ are defined in the polar coordinate system (r, θ) as

$$M_{s,m} = \int_0^{2\pi} \int_0^{\infty} r^s f(r,\theta) e^{-jm\theta} r\,dr\,d\theta \tag{5.5}$$

where $f(r, \theta)$ is the image and $m = 0, +1, +2, \ldots$ is the circular harmonic order. By definition, the Mellin transform order s is complex valued. With integer $s \geq 0$, OFM moments now can be defined as:

$$\phi_m = \frac{1}{2\pi a_n} \int_0^{2\pi} \int_0^{1} f(r,\theta) e^{-jm\theta} r\,dr\,d\theta \tag{5.6}$$

where a_n is a normalization constant and $Q_n(r)$ is a polynomial in r of degree n. The set of $Q_n(r)$ is orthogonal over the range $0 \leq r \leq 1$:

$$\int_0^{1} Q_n(r) Q_k(r) r\,dr = a_n \delta_{nk} \tag{5.7}$$

where δ_{nk} is the Kronecker symbol and $r = 1$ is the maximum size of the objects that can be encountered in a particular application. Hence the basis functions $Q_n(r)\ e^{-jm\theta}$ of the OFMM are orthogonal over the interior of the unit circle. OFMM can be thought of as generalized ZM. They have a single orthogonal set of the radial polynomials $Q_n(r)$ for all circular harmonic order q, while ZM have one orthogonal set of radial polynomials $R_p^{|q|}(r)$ for each different circular order q.

ZM focus on the global features and catch less local information than OFMM. Rotation, translation and scale invariance can be obtained in the same way as with Zernike moments given by Kan and Srinath [170].

The reconstructed image from OFMMs can be obtained by

$$\hat{f}(r,\theta) = \sum_{n=0}^{N} \sum_{m=-M}^{M} \phi_{nm} Q_n(r) e^{jm\theta} \tag{5.8}$$

Because of the orthogonality of the set $Q_n(r)\, e^{jm\theta}$, each OFMM makes an independent contribution to the reconstructed image.

As various feature extraction methods are reported in the literature, we have tried to select the best and appropriate features for degraded Gurmukhi text recognition system with experimentation evaluation.

5.3 Performance Analysis of Structural and Statistical Features

The structural features for preparing the feature vector are selected keeping in mind the shape variations of the characters in degraded printed Gurmukhi script. These features are robust to the noise and the features are retained by most of the characters in the database. These features exist even in clean printed Gurmukhi characters.

We have applied different segmentation algorithms on sub-symbols in each zone. Similarly, we have applied different features for sub-symbols of different zones. Table 5.3 contains the set of features applied in different zones.

Table 5.3: Set of different features used for different zones

Zone	features used
Upper	St6 to St12, zoning, ZM, OFMM
Middle	St1 to St12, zoning, ZM, OFMM
Lower	St6 to St12, zoning, ZM, OFMM

5.3.1 Standard feature values of middle zone sub-symbols

Table 5.4 contains standard structural feature values for features St1 to St6 for all Gurmukhi sub-symbols in middle zone. Table 5.5 contains left and right profile chain codes while Table 5.6 contains top and bottom profile chain codes for Gurmukhi sub-symbols in middle zone. Two vowels ੀ, ੈ and the second component of ਗ਼ contain the same shape 'ਾ' falling in middle zone. Therefore, we have considered sub-symbol 'ਾ' for feature extraction. Similarly, for the multicomponent character ਗ਼, we have considered only subsymbol ਰ for experimental purposes. Also, character ੳ contains sub-symbol ੜ in middle zone. Later on, after merging of sub-symbols the vowels ੀ, ੈ and the characters ਗ, ਗ਼ and ੳ will attain their original shape.

Table 5.4: Feature values for Gurmukhi sub-symbols in middle zone for features St1 to St6

Char.	St1	St2	St3	St4	St5	St6
ਚ	0	0	1	2	1	0.926
ਅ	1	0	0	0	2	0.872
ਏ	0	1	1	2	1	0.975
ਸ	1	0	1	2	2	0.974
ਹ	1	0	1	1	1	0. 948
ਕ	0	0	1	1	1	0.975
ਖ	1	1	1	0	1	0.950
ਘ	1	0	0	0	2	0.886
ਙ	0	0	1	1	1	0.974
ਚ	1	0	1	1	1	1.000
ਛ	0	1	1	1	1	0.973
ਜ	1	0	1	1	2	0.974
ਝ	0	0	1	2	1	1.026
ਞ	0	1	1	2	1	0.974
ਟ	0	1	1	1	1	0.974
ਠ	0	0	1	1	1	0.974
ਡ	0	0	1	1	1	1.000
ਢ	0	1	1	1	1	1.026
ਣ	0	0	1	1	1	0.974
ਤ	0	0	1	1	1	0.948
ਥ	1	0	1	2	1	1.000
ਦ	0	1	1	1	1	0.974
ਧ	1	0	1	2	1	0.973
ਨ	0	0	1	1	2	0.973
ਪ	1	0	0	0	1	0.974
ਫ	0	1	1	1	1	1.000
ਬ	1	0	1	2	1	0.950
ਭ	0	0	1	1	1	0.948
ਮ	1	0	0	0	2	1.000
ਯ	1	0	1	2	2	0.975
ਰ	1	0	1	1	1	1.081
ਲ	0	0	1	2	2	0.951
ਵ	0	1	1	1	1	0.974
ੜ	0	0	1	1	1	1.000
ਸ਼	1	0	1	1	1	0.974
ਜ਼	1	0	1	1	2	0.974
ਖ਼	1	0	0	0	1	0.950
ਫ਼	0	1	1	1	1	1.026
ੁ	1	0	1	1	1	1.081
ਲ਼	0	0	1	2	2	0.976
ਾ	0	1	1	2	0	3.455
ਾ	1	0	1	1	1	3.455

Table 5.5: Left and right profile chain codes for Gurmukhi sub-symbols in middle zone

Char	Left profile chain codes (St7)			Right profile chain codes (St8)		
	East	South	West	East	South	West
ੳ	41.67	58.33	0.00	18.18	30.91	50.91
ਅ	56.16	35.62	8.22	2.22	82.22	15.56
ੲ	49.02	33.33	17.65	28.73	25.29	45.98
ਸ	50.77	40.00	9.23	32.74	29.21	38.05
ਹ	45.12	28.05	26.83	0.0	68.89	31.11
ਕ	56.07	15.89	28.04	27.78	36.11	36.11
ਖ	51.67	41.67	6.66	2.22	77.78	20.0
ਘ	54.55	39.39	6.06	33.33	27.78	38.89
ਙ	53.06	21.43	25.51	32.94	25.88	41.18
ਚ	45.65	25.00	29.35	0.0	69.57	30.43
ਛ	40.70	29.07	30.23	29.63	28.40	41.97
ਜ	37.50	32.95	29.55	0.0	86.05	13.95
ਝ	52.69	16.13	31.18	34.51	17.70	47.79
ਞ	52.94	33.33	13.73	34.55	20.00	45.45
ਟ	50.62	17.28	32.10	30.87	19.75	49.38
ਠ	46.88	28.12	25.00	20.90	31.34	47.76
ਡ	45.38	14.28	40.34	35.59	14.41	50.00
ਢ	49.47	23.16	27.37	25.30	33.74	40.96
ਣ	52.31	18.46	29.23	36.28	15.93	47.79
ਤ	48.80	12.00	39.20	16.00	34.00	50.00
ਥ	50.85	40.68	8.47	25.58	40.70	33.72
ਦ	48.91	20.65	30.44	37.82	15.38	46.80
ਧ	51.72	39.66	8.62	2.33	76.74	20.93
ਨ	40.68	35.59	23.73	19.35	37.10	43.55
ਪ	52.54	38.93	8.47	0.0	81.82	18.18
ਫ	47.78	24.44	27.78	39.13	13.66	47.21
ਬ	55.38	32.31	12.31	2.32	81.40	16.28
ਭ	48.33	13.33	38.34	6.52	50.00	43.48
ਮ	51.47	38.24	10.29	11.36	61.37	27.27
ਯ	53.23	40.32	6.45	0.0	86.05	13.95
ਰ	50.00	22.73	27.27	34.21	20.18	45.61
ਲ	38.77	46.94	14.29	15.38	42.31	42.31
ਵ	49.42	21.84	28.74	33.64	22.43	43.93
ੜ	47.15	12.20	40.65	12.76	48.94	38.30
ਸ਼	50.77	40.00	9.23	32.74	29.21	38.05
ਜ਼	37.50	32.95	29.55	0.00	86.05	13.95
ਖ਼	53.25	28.57	18.18	2.22	77.78	20.00
ਫ਼	38.46	26.37	35.17	34.81	12.15	53.04
ਰ਼	50.00	18.87	31.13	34.21	20.18	45.61
ਲ਼	28.85	44.23	26.92	10.67	29.33	60.00
ਾ	13.64	86.36	0	0	90.48	9.52
ਿ	7.50	92.50	0	0	94.47	5.13

Table 5.6: Top and bottom profile chain codes for Gurmukhi sub-symbols in middle zone

Char	Top profile chain codes (St9)			Bottom profile chain codes (St10)		
	East	South	North	East	South	North
ੳ	88.89	6.67	4.44	18.55	27.42	54.03
ਅ	35.14	32.43	32.43	10.42	35.42	54.17
ੲ	22.77	38.62	38.61	15.39	32.69	51.92
ਸ	92.31	5.13	2.56	18.18	30.52	51.30
ਹ	23.46	38.27	38.27	27.66	27.66	44.68
ਕ	92.50	5.00	2.50	22.34	21.28	56.38
ਖ	63.64	18.18	18.18	20.61	27.48	51.91
ਘ	20.93	39.53	39.54	17.52	29.20	53.28
ਙ	30.14	42.46	27.40	35.72	30.95	33.33
ਚ	46.88	26.56	26.56	24.44	20.00	55.56
ਛ	56.86	21.57	21.57	39.68	4.76	55.56
ਜ	16.66	41.67	41.67	24.03	25.58	50.39
ਝ	86.84	7.90	5.26	18.18	27.27	54.55
ਞ	25.28	37.36	37.36	13.51	28.83	57.66
ਟ	31.04	34.48	34.48	24.14	18.96	56.90
ਠ	18.60	40.70	40.70	26.76	14.08	59.16
ਡ	97.37	2.63	0.00	18.70	27.64	53.66
ਢ	84.21	7.89	7.90	17.50	27.50	55.00
ਣ	16.36	41.82	41.82	16.52	27.83	55.65
ਤ	25.84	37.08	37.08	29.85	11.94	58.21
ਥ	86.84	7.90	5.26	19.68	27.56	52.76
ਦ	92.31	5.13	2.56	15.26	28.81	5.93
ਧ	19.14	40.43	40.43	34.69	28.57	36.74
ਨ	26.98	47.62	25.40	16.47	37.65	45.88
ਪ	38.10	30.95	30.95	17.69	28.46	53.85
ਫ	92.31	5.13	2.56	18.33	27.50	54.17
ਬ	22.50	38.75	38.75	44.18	27.91	27.91
ਭ	36.36	31.82	31.82	30.91	7.27	61.82
ਮ	49.26	25.37	25.37	14.10	32.05	53.85
ਯ	20.72	39.64	39.64	19.05	24.76	56.19
ਰ	86.49	8.11	5.40	14.41	28.81	56.78
ਲ	90.24	4.88	4.88	8.82	35.88	55.30
ਵ	53.48	23.26	23.26	20.59	23.53	55.88
ੜ	24.48	37.76	37.76	17.57	18.92	63.51
ਸ਼	92.31	5.13	2.56	13.48	34.27	52.25
ਜ਼	16.66	41.67	41.67	17.95	30.77	51.28
ਖ਼	63.64	18.18	18.18	19.18	29.45	51.37
ਫ਼	92.31	5.13	2.56	16.91	29.41	53.68
ਰ਼	86.49	8.11	5.40	13.33	31.11	55.56
ਲ਼	90.24	4.88	4.88	13.04	28.70	58.26
ਾ	100	0	0	100	0	0
ਿ	100	0	0	100	0	0

5.3.2 Standard feature values of upper and lower zone sub-symbols

Table 5.7 contains standard structural feature values for features St6 for all Gurmukhi sub-symbols in upper zone and lower zone. Table 5.8 contains left and right profile chain codes while Table 5.9 contains top and bottom profile chain codes for Gurmukhi characters in upper and lower zones. Two vowels ੀ, ਿ contains the same shape ‘ ◠ ’ falling in upper zone. Also symbol ੲ contains sub-symbol ◠ in upper zone and symbol ੳ contains sub-symbol ⌐ in upper zone. We have considered all these sub-symbols in upper zone under the same class. Later on, after merging of sub-symbols, the symbols will attain their original shape.

Table 5.7: Standard structural feature values for feature St6 for Gurmukhi sub-symbols in upper and lower zone

Sub-symbol	Feature St6
Upper zone sub-symbols	
◝	0.625
◝	0.667
◝	0.687
◝	0.600
◝	1.000
◝	0.526
◝	0.437
Lower zone sub-symbols	
▁	0.278
▁	0.579
▁	0.769
▁	0.833
▁	1.077

Table 5.8: Left and right profile chain codes for Gurmukhi sub-symbols in upper and lower zone

Char	Left profile chain codes (St7)			Right profile chain codes (St8)		
	East	South	West	East	South	West
Upper zone sub-symbols						
◌	82.35	5.88	11.77	42.86	14.28	42.86
◌	72.00	12.00	16.00	42.86	14.28	42.86
◌	71.43	23.81	4.76	39.39	21.22	39.39
◌	58.82	41.18	0.0	41.86	13.95	44.19
◌	62.50	12.50	25.00	14.28	42.86	42.86
◌	38.89	22.22	38.89	41.18	17.64	41.18
◌	75.00	12.50	12.50	41.18	5.88	52.94
Lower zone sub-symbols						
◌	62.50	37.50	0.00	0.00	37.50	62.50
◌	55.55	16.67	27.78	40.00	8.00	52.00
◌	65.52	10.34	24.14	38.46	23.08	38.46
◌	54.17	8.33	37.50	30.00	30.00	40.00
◌	54.84	12.90	32.26	39.47	18.42	42.11

Table 5.9: Top and bottom profile chain codes for Gurmukhi sub-symbols in upper and lower zone

Char	Top profile chain codes (St9)			Bottom profile chain codes (St10)		
	East	South	North	East	South	North
Upper zone sub-symbols						
◌	47.37	47.37	5.26	30.44	34.78	34.78
◌	57.89	42.11	0.00	50.00	25.00	25.00
◌	61.11	33.33	5.56	40.74	29.63	29.63
◌	71.43	23.81	4.76	27.50	35.00	37.50
◌	28.57	42.86	28.57	28.57	28.57	42.86
◌	31.03	48.28	20.69	44.00	28.00	28.00
◌	62.50	18.75	18.75	38.10	23.80	38.10
Lower zone sub-symbols						
◌	77.78	11.11	11.11	68.42	10.53	21.05
◌	73.68	15.79	10.53	70.00	10.00	20.00
◌	31.25	56.25	12.50	25.00	37.50	37.50
◌	30.44	34.78	34.78	37.50	25.00	37.50
◌	60.00	20.00	20.00	20.83	29.17	50.00

5.3.3 Performance analysis of features for middle zone sub-symbols

Table 5.10 contains the possible values of first six structural features along with the accuracies with which these features have been calculated.

Table 5.10: Analysis of structural features from St1 to St6

Feature	Percentage accuracy of detection of feature	Number of features used	Possible values (true = 1, false = 0)
St1	96	1	0 or1
St2	94	1	0 or1
St3	100	1	0 or1
St4	89	1	0 or1
St5	97	1	0 or1
St6	98	1	0 or 1 or 2

We have also analyzed the performance of other features. The performance analysis of these features has been done using *k*-NN classifier. We have used $k = 1$ and MATLAB 7.2 has been used for the experimental purpose. Table 5.11 contains the percentage accuracy of structural features St7 to St10 for thinned and unthinned data for recognition of Gurmukhi sub-symbols in middle zone. Similarly, Table 5.12 contains percentage accuracy of structural feature St11, Table 5.13 contains percentage accuracy of structural feature St12 and Table 5.14 contains percentage accuracy of zoning feature for thinned and unthinned data for recognition of Gurmukhi sub-symbols in middle zone. Furthermore, Table 5.15 contains percentage accuracy of Zernike moments and Table 5.16 contains percentage accuracy of OFM moments for unthinned data for recognition of Gurmukhi sub-symbols in middle zone.

Table 5.11: Percentage accuracy of structural feature St7 to St10 for middle zone sub-symbols

Feature name	Number of feature values	Percentage accuracy
St7 to St10(unthinned)	12	18.40
St7 to St10(thinned)	12	10.94

Table 5.12: Percentage accuracy of structural feature DDD for middle zone sub-symbols

Number of feature values	Percentage accuracy	
	Thinned	Unthinned
256(all 8 directions)	47.34	69.65
128 (4 even directions)	41.78	60.28
128 (4 odd directions)	38.60	58.70

Table 5.13: Percentage accuracy of transitions feature for middle zone sub-symbols

Number of features values	Value of *M* (maximum transitions)	Value of *T*	Percentage accuracy	
			Thinned	Unthinned
200	5	10	48.54	73.63
140	5	7	43.81	66.66
100	5	5	45.27	66.17
160	4	10	49.37	74.12
112	4	7	41.39	67.16
80	4	5	34.21	66.67
120	3	10	42.49	73.63
84	3	7	39.93	68.66
60	3	5	37.05	65.67

Table 5.14: Percentage accuracy of zoning feature for middle zone sub-symbols

Number of features values	Grid size ($N \times N$)	Percentage accuracy	
		Thinned	Unthinned
144	4	43.78	72.64
100	5	51.33	74.12
64	6	50.24	72.14
49	7	46.79	68.16
36	8	43.28	63.68
25	10	36.31	58.20
16	12	31.84	43.78

Table 5.15: Percentage accuracy of Zernike moments feature for middle zone sub-symbols

Order of Zernike moments	Number of feature values	Percentage accuracy
16	81	46.23
17	90	48.71
18	100	47.81
19	110	48.80
20	121	47.31
21	132	49.70
22	144	47.81
23	156	47.81
25	182	48.20

Table 5.16: Percentage accuracy of OFMM feature for middle zone sub-symbols

Order of moments	Number of feature values	Percentage accuracy
7	36	60.24
8	45	61.34
9	55	62.19
10	66	60.74
11	78	67.17
12	91	60.67
13	105	62.19
14	120	60.24

It can be seen from the Tables 5.11-5.14 that unthinned data shows better percentage accuracy than thinned data. Also, Figure 5.8 contains the percentage accuracy of structural features St11 for thinned and unthinned data for recognition of Gurmukhi sub-symbols in middle zone. Similarly, Figures 5.9a-5.9c contain percentage accuracy of structural feature St11 for different value of *M* and Figure 5.10 contains percentage accuracy of zoning feature for thinned and unthinned data for recognition of Gurmukhi sub-symbols in middle zone. Figure 5.11 contains percentage accuracy of Zernike moments and Figure 5.12 contains percentage accuracy of OFM moments for unthinned data for recognition of Gurmukhi sub-symbols in middle zone.

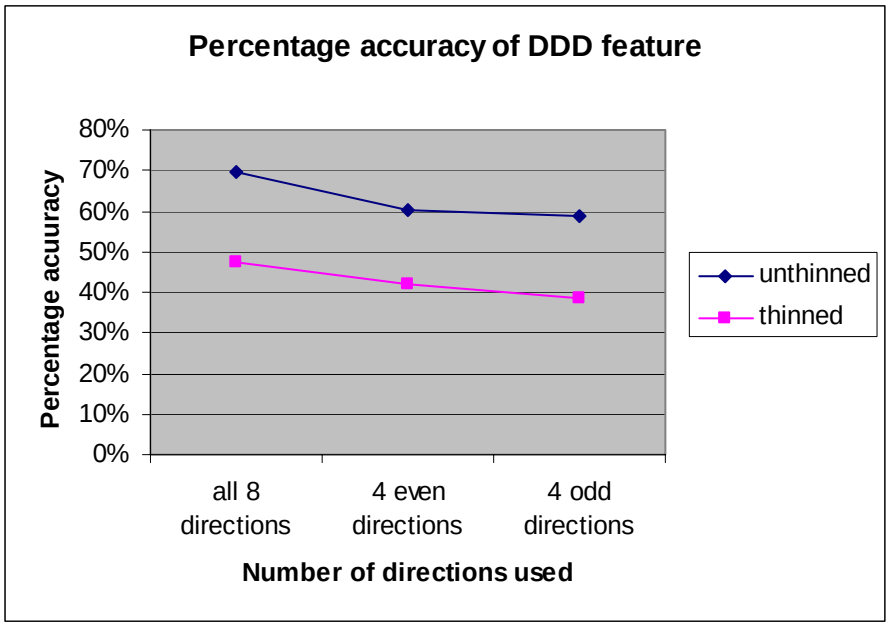


Figure 5.8: Percentage accuracy of DDD structural feature (St11) for sub-symbols in middle zone

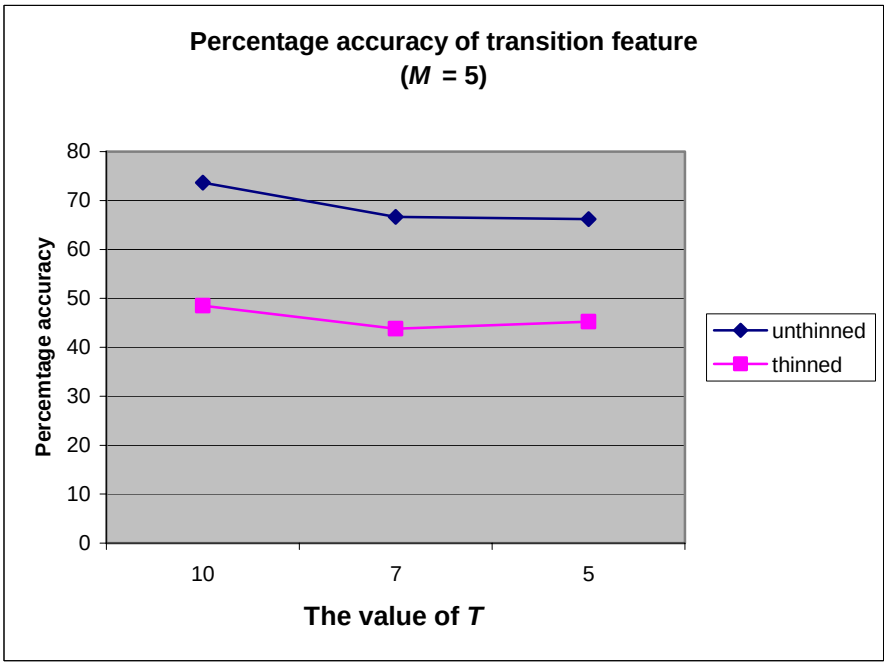


Figure 5.9a: Percentage accuracy of transition feature for sub-symbols in middle zone for different values of *T* having *M* = 5

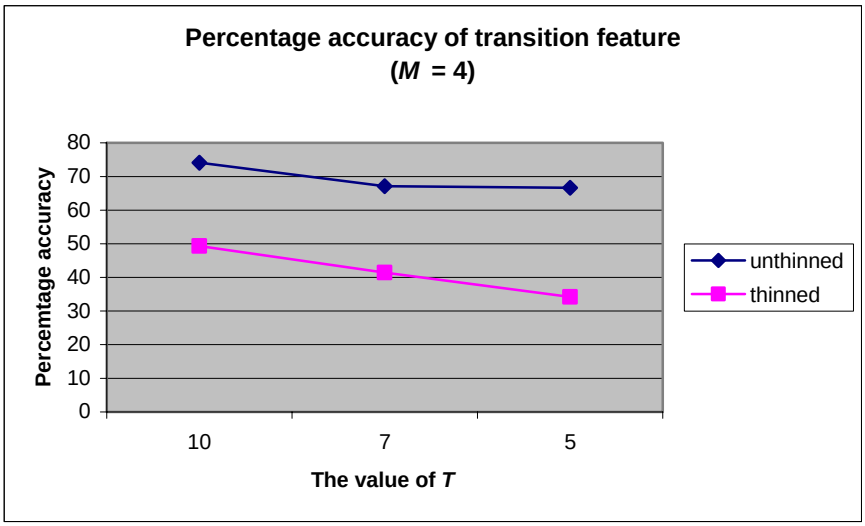


Figure 5.9b: Percentage accuracy of transition feature for sub-symbols in middle zone for different values of *T* having *M* = 4

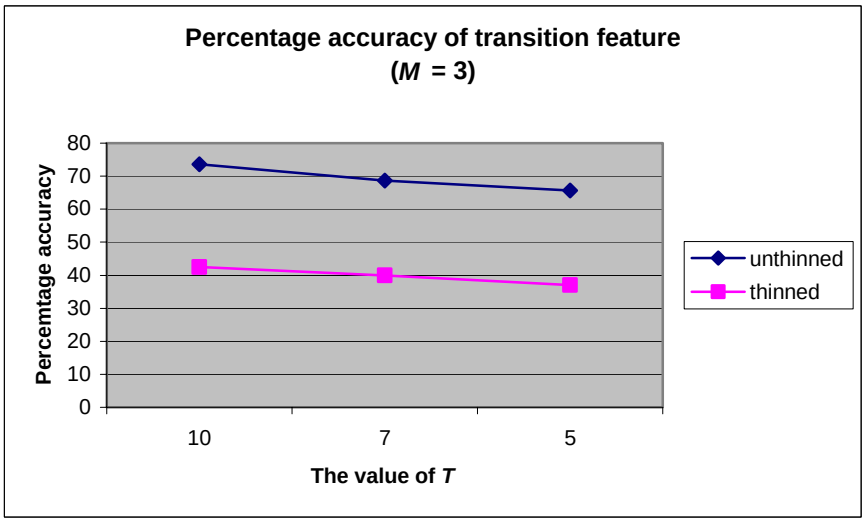


Figure 5.9c: Percentage accuracy of transition feature for sub-symbols in middle zone for different values of *T* having *M* = 3

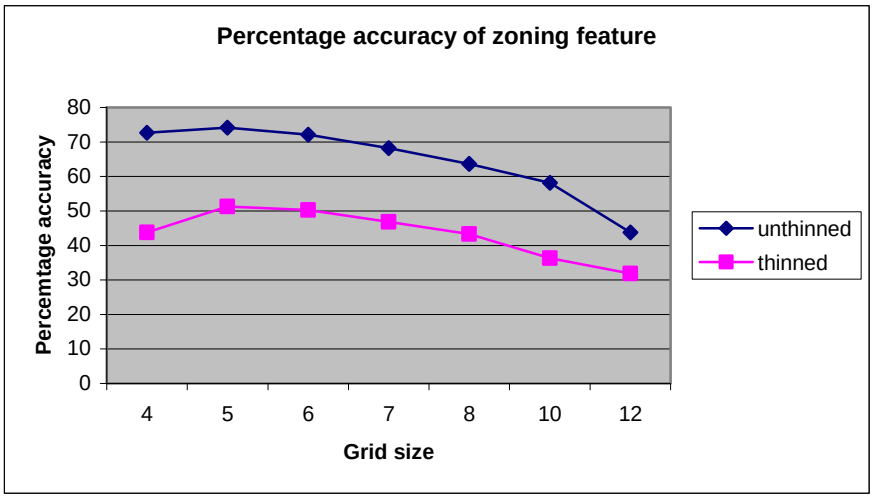


Figure 5.10: Percentage accuracy of zoning feature for sub-symbols in middle zone for different grid sizes

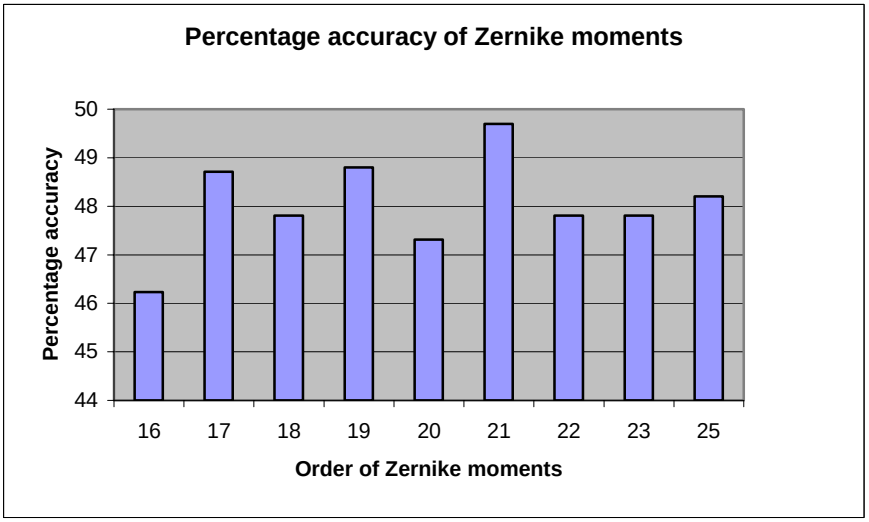


Figure 5.11: Percentage accuracy of Zernike moments feature for sub-symbols in middle zone for different order of Zernike moments

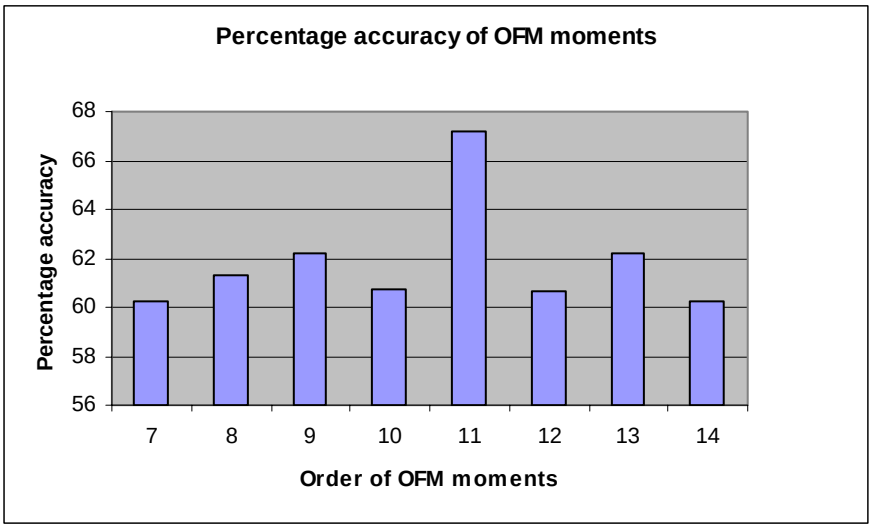


Figure 5.12: Percentage accuracy of OFM moments feature for sub-symbols in middle zone for different order of OFM moments

5.3.4 Performance analysis of features for upper and lower zone characters

Table 5.17 contains the percentage accuracy after applying different features on upper and lower zone characters.

Table 5.17: Percentage accuracies of different features on Gurmukhi sub-symbols in upper and lower zone

Feature	Number of features	Percentage accuracy	
		Upper zone	Lower zone
St6 to St10	13	23.98	16.29
St11	256	80.81	71.37
St11	128 (odd)	79.86	76.21
St11	128 (even)	79.13	70.48
St12	200 (M = 5, T = 5)	75.54	61.23
St12	160 (M = 4, T = 5)	73.38	62.43
St12	120 (M = 3, T = 5)	73.62	61.23
Zoning	144 (grid size = 4)	72.42	57.27
Zoning	100 (grid size = 5)	74.34	58.59
Zoning	64 (grid size = 6)	73.38	56.39
Moments(ZM)	88 (order = 17)	45.56	34.80
Moments(ZM)	130 (order = 21)	46.52	36.45
Moments(ZM)	180 (order = 25)	32.13	32.15
Moments(OFM)	53 (order = 9)	56.35	42.29
Moments(OFM)	76 (order = 11)	60.43	44.49
Moments(OFM)	103 (order = 13)	41.00	41.41

5.4 Size of Feature Set

We have analyzed different structural and statistical features for various options and for thinned and unthinned data. We have made the following observations by analyzing the performance of different features:

(a) Structural features (St1 to St10): The feature vector size for structural features St1 to St10 should be fixed to 18.

(b) DDD feature (St11): We have used three options for this feature for both thinned and unthinned data: taking all the 8 directions (0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, 315^0), taking only even directions (0^0, 90^0, 180^0, 270^0) and taking only odd directions (45^0, 135^0, 225^0, 315^0). It is observed that taking all the eight directions produces better results. Therefore, we have used 256 values for this feature.

(c) Transition feature (St12): Both for thinned and unthinned data, we have used three different options 3, 4 and 5 as maximum transitions (*M)* and for each value of *MAX* three different options 5, 7 and 10 of *T* have been used for this feature. We have analyzed that by taking $M = 4$ and by taking $T = 10$ produces better accuracy for this feature.

(d) Zoning: We have used different grid sizes ranging from 4 to 12 for checking the performance of this feature on degraded printed Gurmukhi data considering thinned and unthinned data separately. A grid size of 5 will divide character image (50×50) into 100 equal sized (5×5) zones. For each of these 100 zones; the average black pixel count is computed giving a feature vector of length 100. this grid size has been considered as it gives better accuracy.

(e) Zernike moments: Zernike moments of different order have been calculated for unthinned data. With experiments, it has been found that as we increase order of moment from 21, the accuracy starts decreasing and best results are obtained at 21 order moments. We obtained total of 132 feature vector set, but first two features are ignored as scale and translation invariance stage does affect two of these Zernike features as explained by Khotanzad [169].

(f) Orthogonal Fourier moments: We have used OFM moments of different order on unthinned data. OFM moments of order 11 produce better accuracy. The feature vector size for this feature thus becomes 78.

The best results produced by a feature for some particular option does not guarantee that while using the combination of the features, the same option will produce good results, as discussed in next chapter. Therefore, the decision of optimal size of the combined feature vector size has been deferred to Chapter 6, where we have analyzed the different results when feature combination is used.

The determination of optimized subset of features used for decision making in classification is an important issue. There are many sophisticated ways like neural networks, genetic algorithm, fuzzy sets or hybrid of these used in literature [175] for optimal selection of feature set. Some features may be redundant or irrelevant, and therefore instead may serve primarily as a source of confusion. It is not necessary true that a large number of features provide better results. Inclusion of irrelevant features increases noise and computational complexity. A method for automated feature selection, includes (a) generating one or more initial sets of features and evaluating the initial sets to determine quality scores for the initial feature sets, (b) choosing selected ones of the feature sets according to the quality scores and modifying the selected feature sets to generate a generation of modified feature sets, (c) evaluating the modified feature sets to determine updated quality scores for the modified feature sets, and (d) repeating (b) and (c) until a modified feature set is satisfactory.

5.5 Discussion

We have analyzed the performance of different structural and statistical features for recognition of degraded Gurmukhi text. The statistical features are invariant to image transformations including translation, rotation, and resolutions / size of the image. The structural features are invariant to noise and shape variations due to touching blobs and heavy printing of the characters. In this chapter, we have decided the values of parameters for different features that give the best results.

The structural features like presence of sidebar, presence of half sidebar, presence of headline, number of junctions with headline, number of junctions with baseline, aspect ratio, left and right profile direction codes, top and bottom profile direction codes and transition features have been used. Another useful structural feature named Directional Distance Distribution has been used, which is based upon the distance of nearest black/white pixel in eight directions for each white/black pixel in the input binary array. Transitions feature and DDD features has been used for first time for recognition of an Indian script. One can see from Table 5.12 and Table 5.13 that these two features have acceptable accuracy.

Statistical features including zoning, Zernike moments, OFMM have been used for extracting features. Zoning feature produces good results at the grid size of 5.

Chapter 6
Classification

Classification stage uses the features extracted in the feature extraction stage to identify the text segment according to the preset rules. We have already discussed the different types of classifiers used for recognition purpose in Chapter 2. Template matching, syntactic or structural methods, statistical methods, artificial neural networks, kernel methods and hybrid classifiers are notably used classification methods. A good amount of literature has been found on the use of these classifiers for different kinds of recognition purposes from printed to handwritten text.

Sometimes calculation of features is very expensive and rather than calculating all features in a batch mode, we calculate features one at a time and at each step we decide, depending on our confidence obtained so far, whether to make another measurement or whether to make a final decision. This is referred to as sequential classification**.** Sometimes, it is impossible to decide the class membership of a pattern without looking at the context in which the original pattern is embedded. In such situations, it is necessary to use contextual information at the classification stage known as contextual classification**.**

We start with the introduction to various classifiers used for experimental purposes.

6.1 Classifiers

We have implemented *k*-Nearest Neighbor, Support Vector Machines and Neural Network classifiers for the experimental purpose for recognition of degraded printed Gurmukhi characters. An introduction to these classifiers and the experimental results on the database has been given in the following sections using three classifiers.

6.1.1 Neural networks

Neural Networks provide a promising approach to solve problems which are very hard for classical approaches. In fact, due to their generalization capability and noise insensitivity, they have been applied to pattern recognition with very encouraging results. We have performed all the neural network experiments using the NeuNet Pro 2.3 software.

An SFAM is a version of the fuzzy ARTMAP neural network model [174]. SFAM was designed to improve the computational efficiency of the fuzzy ARTMAP model with a minimal loss of learning effectiveness. The "fuzzy" component in the name of this network refers to the fact that its learning process implements fuzzy logic operations in order to achieve a number of key pattern matching and adaptation functions [174]. The general architecture of SFAM neural network has been shown in Figure 6.1.

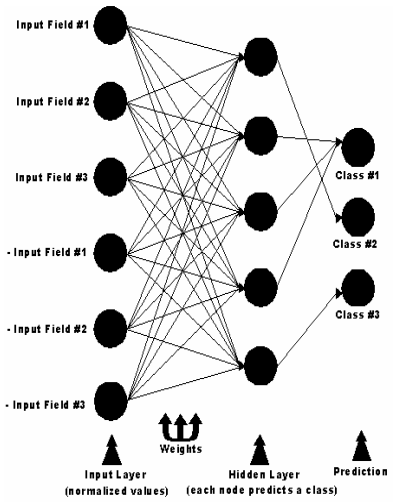


Figure 6.1: SFAM Neural network

SFAM has mainly two layers-input and output. Input to network flows through the complement coder, which normalizes input string and expands to twice its original size by adding its complement. Complement coded input then flows into the input layers and remains there. Weights (W) from each of the output category nodes flow down to input layer. Category layer merely holds the names of M number of categories that the network has to learn. Vigilance parameter and match tracking are mechanisms of the network. Vigilance parameter (in the range (0, 1)) controls granularity of output node. Vigilance decides on

whether a particular output node is good enough to encode a given input pattern or whether a new output node should be opened to encode the same.

When error occurs in training phase during classification of patterns, *i.e.*, when selected output node does not represent the same output category corresponding to input pattern presented, then match tracking is evoked, which may result in network adjusting its learning parameters and network can open new output nodes. Vigilance decides on whether a particular output node is good enough to encode a given input pattern or whether a new output node should be opened to encode the same.

Network is said to be in the state of resonance if the match function value exceeds vigilance parameter. For a node to exhibit resonance, it is essential that it not only encodes the given input pattern, but should also represent the same category as that of input pattern.

Network is said to be in the state of mismatch reset if vigilance parameter exceeds match function. It means that particular output node is not fit enough to learn the given input pattern and so cannot update its weight.

As a way of illustration, Figures 6.2a-6.2c depict a number of situations during the learning process of a SFAM. Figure 6.3a illustrates the initial state of a SFAM before learning to classify two categories, $C1$ and $C2$. In this case, there are no nodes represented in the output layer until the network has its first opportunity to “learn” a pattern. Once an input pattern is presented, an output node is formed to represent it (Figure 6.2b). Such an output node is linked to the category label indicated in the category layer. The matching and/or creation of output nodes as well as the adaptation of weights are based on the steps outlined above [174]. After a number of learning steps, the network consists of a number of output nodes that encode a number of input patterns. Figure 6.2c illustrates this situation without showing the weight connections of nodes $O2$ and $O3$. The output nodes can also be seen as sub-classes of the taught categories $C1$ and $C2$ (Figure 6.2c). For instance, nodes $O1$ and $O2$ encode (cluster) input patterns that belong to category $C1$, while $O3$ encodes patterns that belong to category $C2$.

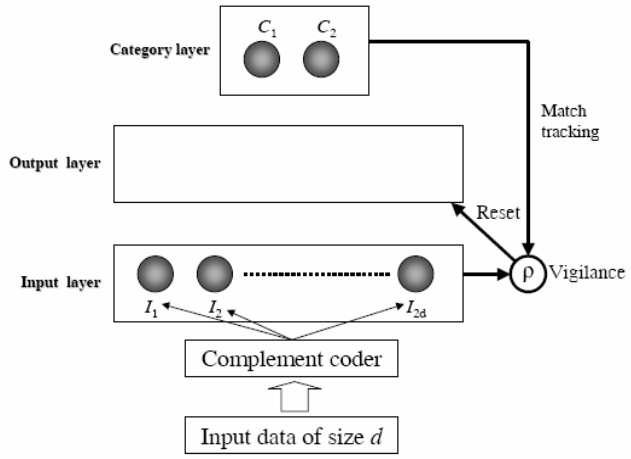


Figure 6.2a: A SFAM neural network before starting a learning process

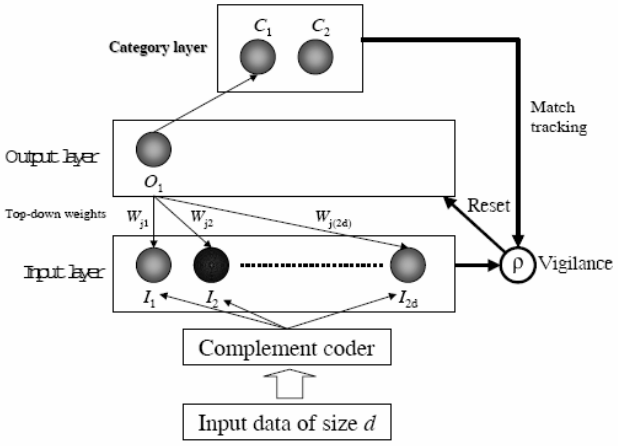


Figure 6.2b: A SFAM neural network after the first input pattern has been learned

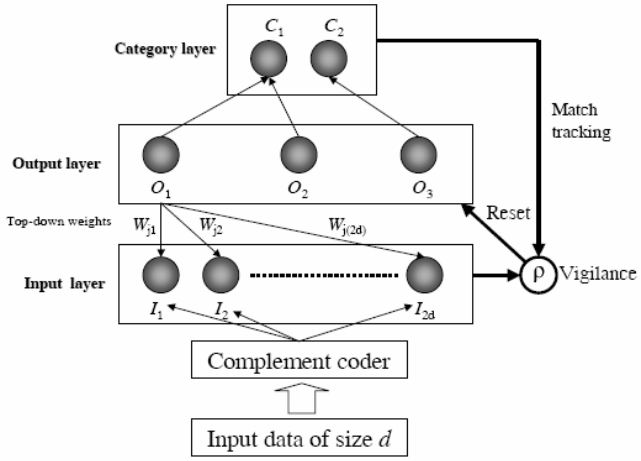


Figure 6.2c: A SFAM neural network after a number of learning steps

6.1.2 Nearest neighbor classifier

We have used Euclidean distance for finding the nearest neighbor. Euclidian distance is the straight line distance between two points in an *n*-dimensional space. The Euclidean distance between an input feature vector *X* and a library feature vector *C* is given by

$$D = \sqrt{\sum_{i=1}^{N} (C_i - X_i)(C_i - X_i)} \tag{6.1}$$

where C_i is the i^{th} library feature and X_i is the i^{th} input feature and N is the number of features used for classification. The class of the library feature vector producing the smallest Euclidean distance, when compared with the library input feature vector, is assigned to the input character. For computational efficiency, only the square of the distance is considered.

The *k*-NN is more general than nearest-neighbor. Putting it other way, nearest-neighbor is a special case of *k*-NN, where $k = 1$. For the tests in this thesis, we have selected $k = 1, 3, 5, 7, 9$ and 11 and compared the results to find out the optimal value of *k*.

6.1.3 SVM

SVM are based on statistical learning theory that uses supervised learning [103]. In supervised learning, a machine is trained instead of programmed, to perform a given task on a number of input-output pairs. According to this paradigm, training means choosing a function which best describes the relation between the inputs and the outputs. The central problem in statistical learning theory is how well the chosen function generalizes, or how well it estimates the output for previously unseen inputs. In general, any learning problem in statistical learning theory will lead to a solution of the type

$$f(x) = \sum_{i=1}^{l} c_i K(x, x_i) \tag{6.2}$$

where x_i , $i = 1, \ldots, l$ are the input examples, *K* a certain symmetric positive definite function named kernel, and c_i a set of parameters to be determined from the examples. One can refer the detailed literature of SVM in [103].

We can describe the working of SVM equivalent to a statistical learning machine that maps points of different categories from *n*-dimensional space into a higher dimensional space

where the two categories are more separable. It tries to find an optimal hyperplane in that high dimensional space that best separates the two categories of points.

Essentially, the hyperplane is learned by the points that are located closest to the hyperplane which are called support vectors. There can be more than one support vector on each side of the plane. Figure 6.3 shows an example of two categories of points separated by a hyperplane.

The limitations of SVM are the selection of a suitable kernel, speed and size, both in training and testing [103].

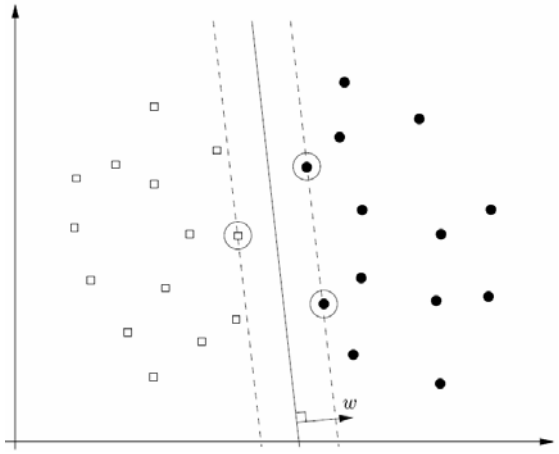


Figure 6.3: Separating hyperplane for feature selection where circles indicate the support vectors

Multiclass SVM involves the construction of binary SVM classifiers for all pairs of classes. In other words, for every pair of classes, a binary SVM problem is solved (with the underlying optimization problem to maximize the margin between two classes). The decision function assigns an instance to a class that has the largest number of votes called as Max Wins strategy. If ties still occur, each sample will be assigned a label based on the classification provided by the furthest hyper plane.

One of the benefits of this approach is that for every pair of classes we deal with a much smaller optimization problem, and in total we need to solve $k(k - 1)/2$ Quadratic Programming (QP) problems of size smaller than n. Given that QP optimization algorithms

used for SVMs are polynomial to the problem size, such a reduction can yield substantial savings in the total computational time.

6.2 Merging Sub-symbols

Merging of sub-symbols is required to convert the recognised sub-symbols to Gurmukhi characters. For merging the sub-symbols, we have used the same technique as used by Lehal [77]. The author has used following rules for merging the sub-symbols.

1. If sub-symbol ◌ is found in the upper zone and sub-symbol ◌ is found in vertically overlapping middle zone below ◌ then merge the 2 sub-symbols to form character ਉ.
2. If sub-symbol ◌ is found in the upper zone and sub-symbol ◌ is found in vertically overlapping middle zone below ◌ and left end of ◌ then combine the 2 sub-symbols to form character ਓ
3. If sub-symbol ◌ is found in the upper zone and if the sub-symbol ◌ is found in vertically overlapping middle zone middle zone below ◌ and on the right end of ◌ then combine the 2 sub-symbols to form character ◌ਾ
4. If sub-symbol ◌ is found in the upper zone and none of the condition 1, 2, or 3 is true then recognise the sub-symbol ◌ as character *tippi* ◌ਂ
5. If sub-symbol ◌ is followed by sub-symbol ◌, in the middle zone and sub-symbol ◌ is the next sub-symbol present in the upper zone and ◌, ◌ and ◌ are overlapping vertically then combine the three sub-symbols ◌, ◌ and ◌ to form characters ਰ and ◌ਾ
6. If sub-symbol ◌ is followed by sub-symbol ◌ in the middle zone and sub-symbol ◌ is the next sub-symbol present in the upper zone and ◌ and ◌ are overlapping vertically then combine the three sub-symbols ◌, ◌ and ◌to form characters ਗ and *tippi* (◌ਂ)
7. If sub-symbol ◌ is followed by sub-symbol ◌, in the middle zone and sub-symbol ◌ is the next sub-symbol present in the upper zone and ◌ and ◌ are overlapping

vertically then combine the three sub-symbols ੍, ਾ and । to form characters ਰ and ਿ

8. If sub-symbol ੜ is followed by sub-symbol । in the middle zone then combine the two sub-symbols to form character ੜਾ
9. If two _ characters are found in vertically overlapping areas in lower zone, they are combined to form the character =
10. If the characters ਸ, ਜ, ਖ, ਫ or ਲ have the sub-symbol " . " present in their lower zone then they are converted to characters ਸ਼, ਜ਼, ਖ਼, ਫ਼ and .ਲ, respectively.

6.3 Implementation details

For neural network experiments, we have used NeuNet Pro version 2.3 for windows, which is a neural network tool. We have also used MATLAB 7.2 for analyzing various results. NeuNet Pro reads the data file only and when used in SFAM classification mode, prediction field may contain up to 256 different classes.

We have used MATLAB 7.2 for performing the experiments for classification, based on *k*-NN classifier. As already discussed in Chapter 5, there exist various options for single feature extraction method. Moreover, combinations of different feature extraction methods can be used. We have created different data files for each option of each different feature method used. For example, to perform an experiment using transition feature (St12) using $M = 5$ and $T = 5$, we have created *train* and *test* data files and for the same feature if $M = 5$ and $T = 7$, we have created different *train* and *test* data files. For performing experiments on the combination of different feature extraction methods, we have combined the data files created by different feature extraction methods and normalized the data of the combined data file to a uniform range. For example, to perform experiments on combined features of DDD (St11) with even directions and transitions feature (St12) using $M = 5$ and $T = 5$, we have created *train* and *test* data files containing combined features of St11 and St12. The combined values of the new data files have also been normalized. If we have two feature sets f_1 and f_2, and range of feature values for f_1 is 0 to m_1 while range of feature values for f_2 is 0 to m_2, the combined data files will be normalized to 0 to m_{12} where m_{12} is the minimum of m_1 and m_2. For each experiment, we have created *train* and *test* files separately using same methods.

We have taken 1269 samples for training and 2211 samples for testing containing 42 different classes in middle zone. Similarly, 245 training samples and 417 testing samples have been selected for testing 9 different sub-symbols appearing in upper zone. Also, we have considered 156 training samples and 227 testing samples for testing 5 sub-symbols appearing in lower zone. For each experiment containing *train* and *test* data file, we have performed followings steps to calculate the percentage accuracy using *k*-NN.

1. Import the *train* and *test* files using *uimport* command in the MATLAB. First column of *train* file is used to create a *Group* matrix containing the target value of each character. Rests of the columns of training file are used to create a matrix *Train*. Each row of *Train* belongs to the group whose value is the corresponding entry of *Group*. A matrix *Test* is created from testing file containing only features of unknown characters. *Test* and *Train* must be matrices with the same number of columns. *Group* classifies the rows of the data matrix *Test* into groups, based on the grouping of the rows of *Train*. For example, to perform an experiment for middle zone sub-symbols using transition feature (St12) using $M = 5$ and $T = 5$ (number of features = 200), dimensions of *Group* file will be 1269×1, that of *Train* file will be 1269×200 and *Test* file will be of dimension 2185×200.
2. Use *Class = knnclassify (Test, Train, Group, k, distance)*. Argument *k* is the number of nearest neighbors used in the classification and by default, it is 1. Argument *distance* specifies the distance metric and by default it is *Euclidean distance*. The function *knnclassify* classifies the rows of the data matrix *Test* into groups, based on the grouping of the rows of *Train*. The function *knnclassify* assigns each row of *Test* to the group for the closest row of *Train*. *Class* indicates which *Group* each row of *Test* has been assigned to, and is of the same type as *Group*.
3. Export the class data matrix.
4. Find out accuracies in all the cases.

We have performed various experiments using default distance as ‘Euclidean’ and then different values for *k* as 1, 3, 5, 7, 9 and 11.

We have again used MATLAB 7.2 for performing the experiments for classification, based on SVM classifier. As already discussed, different data files for training and testing

have been generated. SVM is a two class classifier. We have used it to solve the problem of multiclass degraded printed Gurmukhi script characters. As already discussed, we are working on 42 different characters of Gurmukhi script in middle zone.

As we have 42 characters to train (n = 42) in Gurmukhi script, the contents of the files will be in accordance with that each having number of entries 42(42 – 1)/2 (= 861). We have used *Linear Kernel* function for SVM. Here, we have applied one against one method with voting. In case of 42 Gurmukhi sub-symbols in middle zone we shall have 861 support vectors.

6.4 Experiments

We have scanned documents from newspapers, magazines, books *etc*. at 300 dpi to create a large set of database consisting of degraded printed Gurmukhi script documents. Each document in the database consists of touching characters and heavily printed characters. After applying segmentation process as discussed in Chapter 4, on a document, we got the segmented characters. These segmented characters have been stored as individual files of size 50 × 50 pixels. Hence, a large database consisting of individual segmented Gurmukhi characters has been prepared for experimental purposes.

6.4.1 Sample images

We have collected the individual characters of degraded printed Gurmukhi script for experimental purposes. Some of the degraded characters have been given in Figure 6.4. One can see in Figure 6.4, the large variability in shapes belonging to a single class.

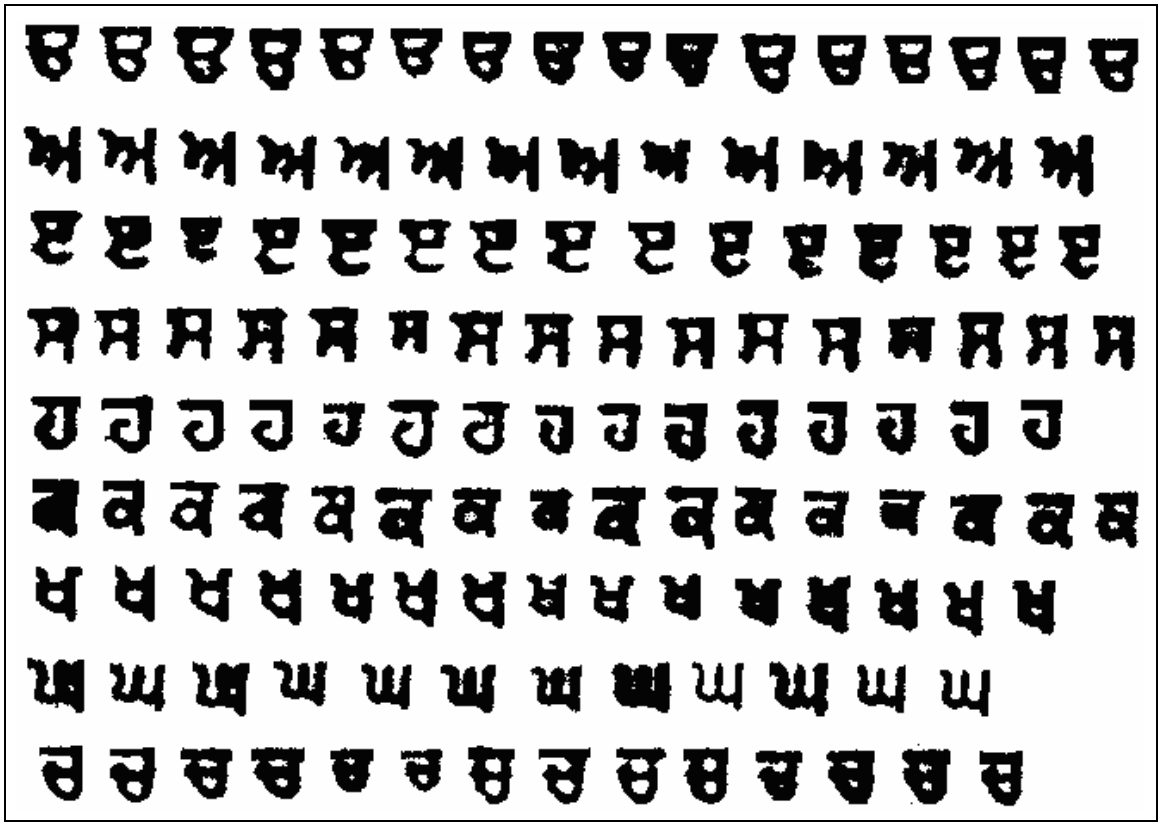


Figure 6.4: Samples of degraded printed Gurmukhi characters for experimental purposes

Two sample documents from the database have already been shown in Figure 1.4 and Figure 1.5. We have taken 3 more sample text images as shown in Figures 6.5-6.7. The recognition accuracy of these figures has been shown in Table 6.1.

Table 6.1: Recognition rate of Gurmukhi text images

Image	Number of Characters	Recognition accuracy (%)
Figure 1.4	887	85.23
Figure 1.5	458	87.33
Figure 6.5	785	84.84
Figure 6.6	1102	84.39
Figure 6.7	855	81.75

The OCR output of Figure 1.4 has been shown in Figure 6.8 and that of Figure 1.5 has been shown in Figure 6.9. The substitution errors are shown in red color and the characters which are missing are shown in green color. There is a large number of touching character pairs in all the three zones in Figure 1.4, but they have been correctly segmented and recognized.

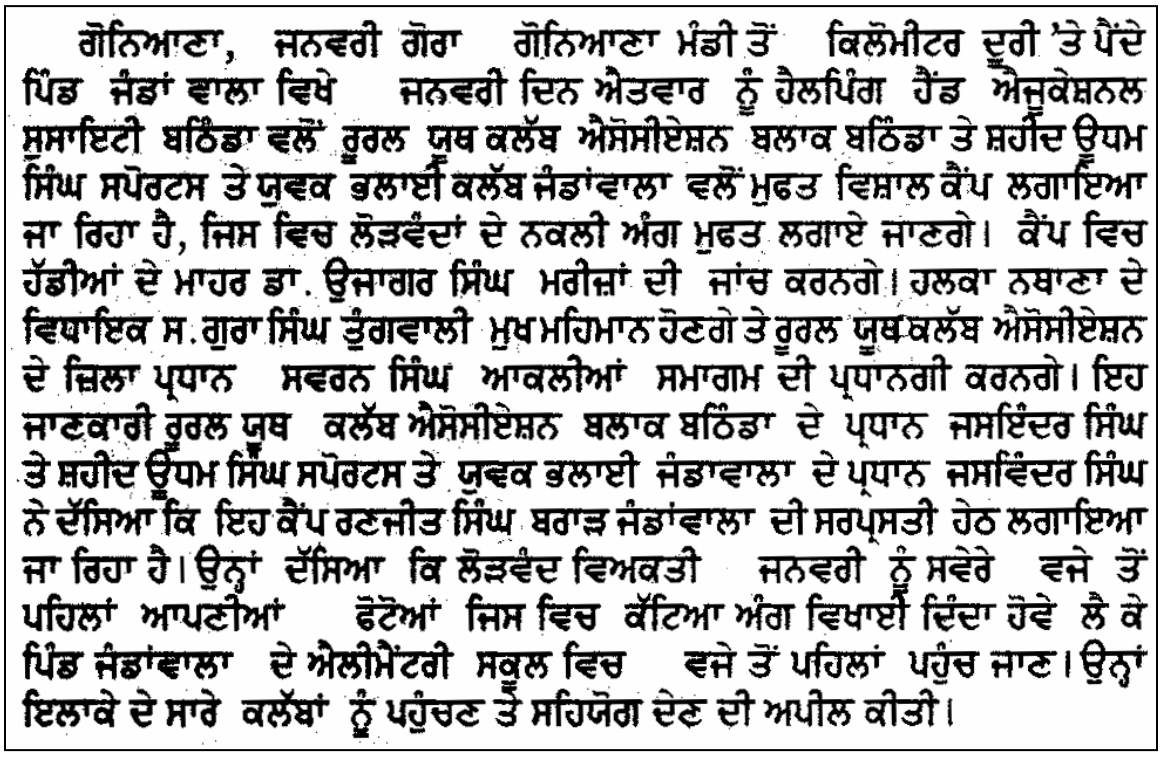
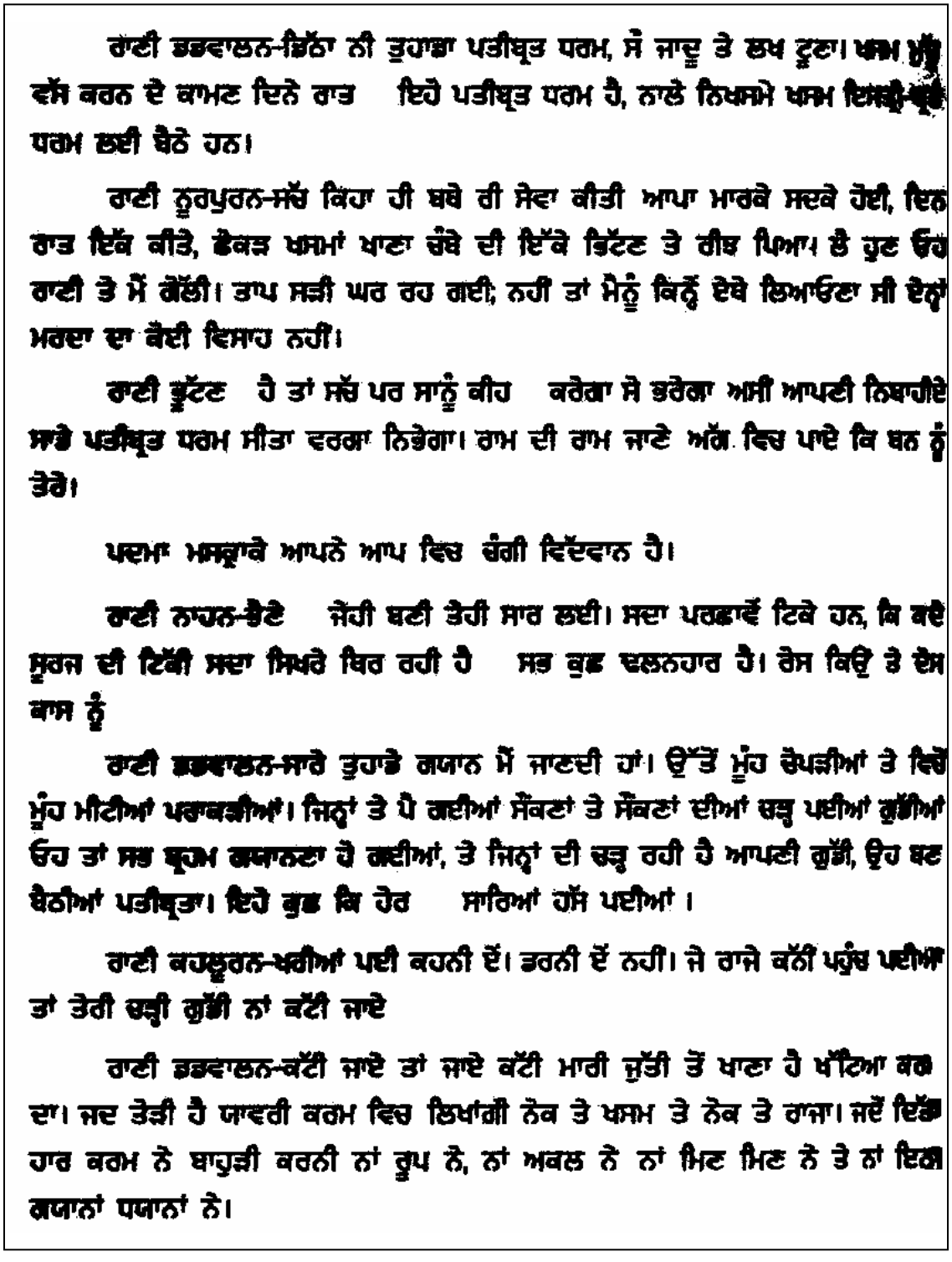

ਗੋਨਿਆਣਾ, ਜਨਵਰੀ ਗੋਰਾ ਗੋਨਿਆਣਾ ਮੰਡੀ ਤੋਂ ਕਿਲੋਮੀਟਰ ਦੂਰੀ 'ਤੇ ਪੈਂਦੇ ਪਿੰਡ ਜੰਡਾਂ ਵਾਲਾ ਵਿਖੇ ਜਨਵਰੀ ਦਿਨ ਐਤਵਾਰ ਨੂੰ ਹੈਲਪਿੰਗ ਹੈਂਡ ਐਜੂਕੇਸ਼ਨਲ ਸੁਸਾਇਟੀ ਬਠਿੰਡਾ ਵਲੋਂ ਰੂਰਲ ਯੂਥ ਕਲੱਬ ਐਸੋਸੀਏਸ਼ਨ ਬਲਾਕ ਬਠਿੰਡਾ ਤੇ ਸ਼ਹੀਦ ਊਧਮ ਸਿੰਘ ਸਪੋਰਟਸ ਤੇ ਯੁਵਕ ਭਲਾਈ ਕਲੱਬ ਜੰਡਾਂਵਾਲਾ ਵਲੋਂ ਮੁਫਤ ਵਿਸ਼ਾਲ ਕੈਂਪ ਲਗਾਇਆ ਜਾ ਰਿਹਾ ਹੈ, ਜਿਸ ਵਿਚ ਲੋੜਵੰਦਾਂ ਦੇ ਨਕਲੀ ਅੰਗ ਮੁਫਤ ਲਗਾਏ ਜਾਣਗੇ। ਕੈਂਪ ਵਿਚ ਹੱਡੀਆਂ ਦੇ ਮਾਹਰ ਡਾ. ਉਜਾਗਰ ਸਿੰਘ ਮਰੀਜ਼ਾਂ ਦੀ ਜਾਂਚ ਕਰਨਗੇ। ਹਲਕਾ ਨਥਾਣਾ ਦੇ ਵਿਧਾਇਕ ਸ. ਗੁਰਾ ਸਿੰਘ ਤੁੰਗਵਾਲੀ ਮੁਖ ਮਹਿਮਾਨ ਹੋਣਗੇ ਤੇ ਰੂਰਲ ਯੂਥ ਕਲੱਬ ਐਸੋਸੀਏਸ਼ਨ ਦੇ ਜ਼ਿਲਾ ਪ੍ਰਧਾਨ ਸਵਰਨ ਸਿੰਘ ਆਕਲੀਆਂ ਸਮਾਗਮ ਦੀ ਪ੍ਰਧਾਨਗੀ ਕਰਨਗੇ। ਇਹ ਜਾਣਕਾਰੀ ਰੂਰਲ ਯੂਥ ਕਲੱਬ ਐਸੋਸੀਏਸ਼ਨ ਬਲਾਕ ਬਠਿੰਡਾ ਦੇ ਪ੍ਰਧਾਨ ਜਸਇੰਦਰ ਸਿੰਘ ਤੇ ਸ਼ਹੀਦ ਊਧਮ ਸਿੰਘ ਸਪੋਰਟਸ ਤੇ ਯੁਵਕ ਭਲਾਈ ਜੰਡਾਵਾਲਾ ਦੇ ਪ੍ਰਧਾਨ ਜਸਵਿੰਦਰ ਸਿੰਘ ਨੇ ਦੱਸਿਆ ਕਿ ਇਹ ਕੈਂਪ ਰਣਜੀਤ ਸਿੰਘ ਬਰਾੜ ਜੰਡਾਂਵਾਲਾ ਦੀ ਸਰਪ੍ਰਸਤੀ ਹੇਠ ਲਗਾਇਆ ਜਾ ਰਿਹਾ ਹੈ। ਉਨ੍ਹਾਂ ਦੱਸਿਆ ਕਿ ਲੋੜਵੰਦ ਵਿਅਕਤੀ ਜਨਵਰੀ ਨੂੰ ਸਵੇਰੇ ਵਜੇ ਤੋਂ ਪਹਿਲਾਂ ਆਪਣੀਆਂ ਫੋਟੋਆਂ ਜਿਸ ਵਿਚ ਕੱਟਿਆ ਅੰਗ ਵਿਖਾਈ ਦਿੰਦਾ ਹੋਵੇ ਲੈ ਕੇ ਪਿੰਡ ਜੰਡਾਂਵਾਲਾ ਦੇ ਐਲੀਮੈਂਟਰੀ ਸਕੂਲ ਵਿਚ ਵਜੇ ਤੋਂ ਪਹਿਲਾਂ ਪਹੁੰਚ ਜਾਣ। ਉਨ੍ਹਾਂ ਇਲਾਕੇ ਦੇ ਸਾਰੇ ਕਲੱਬਾਂ ਨੂੰ ਪਹੁੰਚਣ ਤੇ ਸਹਿਯੋਗ ਦੇਣ ਦੀ ਅਪੀਲ ਕੀਤੀ।

Figure 6.5: Gurmukhi script text image



ਰਾਣੀ ਭਡਵਾਲਨ-ਡਿਠੋਨਾ ਨੀ ਤੁਹਾਡਾ ਪਤੀਬ੍ਰਤ ਧਰਮ, ਸੋ ਜਾਦੂ ਤੇ ਲਖ ਟੂਣਾ। ਖਸਮ ਮੁੰਹ ਵੱਸੇ ਕਰਨ ਦੇ ਕਾਮਣ ਦਿਨੇ ਰਾਤ ਇਹੋ ਪਤੀਬ੍ਰਤ ਧਰਮ ਹੈ, ਨਾਲੇ ਨਿਖਸਮੇ ਖਸਮ ਦਿਸਣ ਖੂਹੇ ਧਰਮ ਲਈ ਬੈਠੇ ਹਨ।

ਰਾਣੀ ਨੂਰਪੁਰਨ-ਸੱਚ ਕਿਹਾ ਹੀ ਬਥੇ ਰੀ ਸੇਵਾ ਕੀਤੀ ਆਪਾ ਮਾਰਕੇ ਸਦਕੇ ਹੋਈ, ਦਿਨ ਰਾਤ ਇੱਕ ਕੀਤੇ, ਡੋਕੜ ਖਸਮਾਂ ਖਾਣਾ ਚੰਬੇ ਦੀ ਇੱਕੋ ਭਿੱਟਣ ਤੇ ਰੀਝ ਪਿਆ। ਲੈ ਚੁਣ ਓਹ ਰਾਣੀ ਤੇ ਮੈਂ ਗੋਲੀ। ਤਾਪ ਸੜੀ ਘਰ ਰਹ ਗਈ; ਨਹੀਂ ਤਾਂ ਮੈਨੂੰ ਕਿਨ੍ਹੇ ਦੇਵੇ ਲਿਆਓਣਾ ਸੀ ਦੇਨ੍ਹਾਂ ਮਰਦਾ ਦਾ ਕੋਈ ਵਿਸਾਹ ਨਹੀਂ।

ਰਾਣੀ ਭੁੱਟੱਣ ਹੈ ਤਾਂ ਸੱਚ ਪਰ ਸਾਨੂੰ ਕੀਹ ਕਰੇਗਾ ਸੋ ਭਰੇਗਾ ਅਸੀਂ ਆਪਣੀ ਨਿਬਾਹੀਏ ਸਾਡੇ ਪਤੀਬ੍ਰਤ ਧਰਮ ਸੀਤਾ ਵਰਗਾ ਨਿਭੇਗਾ। ਰਾਮ ਦੀ ਰਾਮ ਜਾਣੇ ਅੱਗ ਵਿਚ ਪਾਏ ਕਿ ਬਨ ਨੂੰ ਤੋਰੇ।

ਪਦਮਾ ਮਸਕ੍ਰਾਕੇ ਆਪਨੇ ਆਪ ਵਿਚ ਚੰਗੀ ਵਿੱਦਵਾਨ ਹੈ।

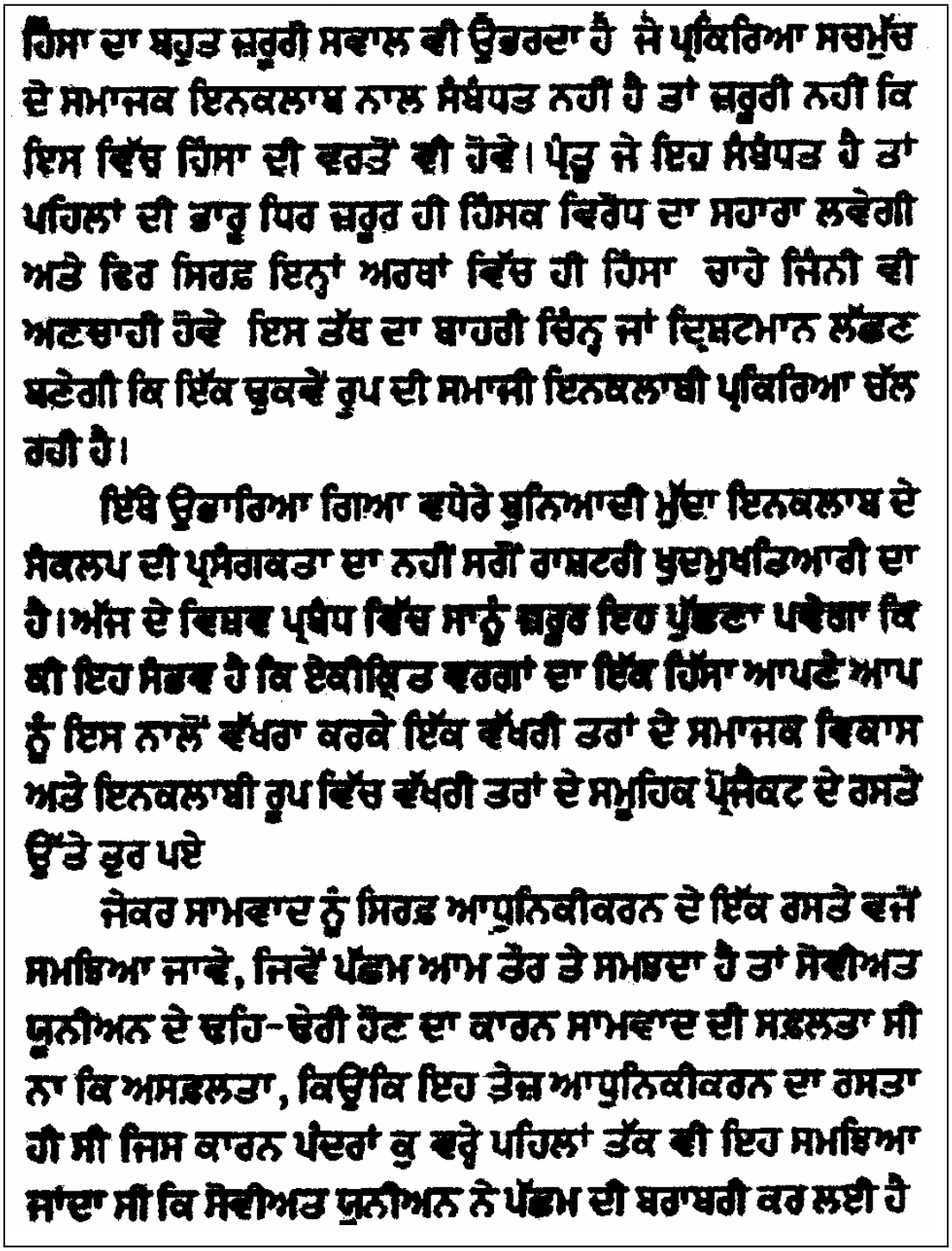
ਰਾਣੀ ਨਾਹਨ-ਭੈਣੇ ਜੋਹੀ ਬਣੀ ਤੈਹੀਂ ਸਾਰ ਲਈ। ਸਦਾ ਪਰਛਾਵੇਂ ਟਿਕੇ ਹਨ, ਕਿ ਕਦੇ ਸੂਰਜ ਦੀ ਟਿੱਕੀ ਸਦਾ ਸਿਖਰੇ ਥਿਰ ਰਹੀ ਹੈ ਸਭ ਕੁਛ ਢਲਨਹਾਰ ਹੈ। ਰੋਸ ਕਿਉ ਤੇ ਦੋਸ ਕਾਸ ਨੂੰ

ਰਾਣੀ ਭਡਵਾਲਨ-ਸਾਰੇ ਤੁਹਾਡੇ ਗਯਾਨ ਮੈਂ ਜਾਣਦੀ ਹਾਂ। ਉੱਤੋਂ ਮੂੰਹ ਚੋਪੜੀਆਂ ਤੇ ਵਿਚੋਂ ਮੂੰਹ ਮੀਟੀਆਂ ਪਰਾਕੜੀਆਂ। ਜਿਨ੍ਹਾਂ ਤੇ ਪੈ ਗਈਆਂ ਸੌਕਣਾਂ ਤੇ ਸੌਕਣਾਂ ਦੀਆਂ ਚੜ੍ਹ ਪਈਆਂ ਕੁੱਡੀਆਂ ਓਹ ਤਾਂ ਸਭ ਬ੍ਰਹਮ ਗਯਾਨਣਾ ਹੋ ਗਈਆਂ, ਤੇ ਜਿਨ੍ਹਾਂ ਦੀ ਚੜ੍ਹ ਰਹੀ ਹੈ ਆਪਣੀ ਕੁੱਡੀ, ਉਹ ਬਣ ਬੈਠੀਆਂ ਪਤੀਬ੍ਰਤਾ। ਇਹੋ ਕੁਛ ਕਿ ਹੋਰ ਸਾਰਿਆਂ ਹੱਸ ਪਈਆਂ।

ਰਾਣੀ ਕਹਲੂਰਨ-ਖਰੀਆਂ ਪਈ ਕਹਨੀ ਏਂ। ਡਰਨੀ ਏਂ ਨਹੀਂ। ਜੇ ਰਾਜੇ ਕੱਨੀਂ ਪਹੁੰਚ ਪਈਆ ਤਾਂ ਤੇਰੀ ਚੜ੍ਹੀ ਕੁੱਡੀ ਨਾਂ ਕੱਟੀ ਜਾਏ

ਰਾਣੀ ਭਡਵਾਲਨ-ਕੱਟੀ ਜਾਏ ਤਾਂ ਜਾਏ ਕੱਟੀ ਮਾਰੀ ਜੁੱਤੀ ਤੋਂ ਖਾਣਾ ਹੈ ਖੱਟਿਆ ਕਰੇ ਦਾ। ਜਦ ਤੇੜੀ ਹੈ ਯਾਵਰੀ ਕਰਮ ਵਿਚ ਲਿਖਾਗੀ ਨੋਕ ਤੇ ਖਸਮ ਤੇ ਨੋਕ ਤੇ ਰਾਜਾ। ਜਦੋਂ ਦਿੱਭ ਹਾਰ ਕਰਮ ਨੇ ਬਾਹੁੜੀ ਕਰਨੀ ਨਾਂ ਰੂਪ ਨੇ, ਨਾਂ ਅਕਲ ਨੇ ਨਾਂ ਮਿਣ ਮਿਣ ਨੇ ਤੇ ਨਾਂ ਇਲਮ ਗਯਾਨਾਂ ਧਯਾਨਾਂ ਨੇ।

Figure 6.6: Gurmukhi script text image


ਹਿੰਸਾ ਦਾ ਬਹੁਤ ਜ਼ਰੂਰੀ ਸਵਾਲ ਵੀ ਉਭਰਦਾ ਹੈ ਜੋ ਪ੍ਰਕਿਰਿਆ ਸਚਮੁੱਚ ਦੇ ਸਮਾਜਕ ਇਨਕਲਾਬ ਨਾਲ ਸੰਬੰਧਤ ਨਹੀਂ ਹੈ ਤਾਂ ਜ਼ਰੂਰੀ ਨਹੀਂ ਕਿ ਇਸ ਵਿੱਚ ਹਿੰਸਾ ਦੀ ਵਰਤੋਂ ਵੀ ਹੋਵੇ। ਪ੍ਰੰਤੂ ਜੇ ਇਹ ਸੰਬੰਧਤ ਹੈ ਤਾਂ ਪਹਿਲਾਂ ਦੀ ਡਾਰੂ ਧਿਰ ਜ਼ਰੂਰ ਹੀ ਹਿੰਸਕ ਵਿਰੋਧ ਦਾ ਸਹਾਰਾ ਲਵੇਗੀ ਅਤੇ ਫਿਰ ਸਿਰਫ਼ ਇਨ੍ਹਾਂ ਅਰਥਾਂ ਵਿੱਚ ਹੀ ਹਿੰਸਾ ਚਾਹੇ ਜਿੰਨੀ ਵੀ ਅਣਚਾਹੀ ਹੋਵੇ ਇਸ ਤੱਥ ਦਾ ਬਾਹਰੀ ਚਿੰਨ੍ਹ ਜਾਂ ਦ੍ਰਿਸ਼ਟਮਾਨ ਲੱਛਣ ਬਣੇਗੀ ਕਿ ਇੱਕ ਚੁਕਵੇਂ ਰੂਪ ਦੀ ਸਮਾਜੀ ਇਨਕਲਾਬੀ ਪ੍ਰਕਿਰਿਆ ਚੱਲ ਰਹੀ ਹੈ।

ਇੱਥੇ ਉਭਾਰਿਆ ਗਿਆ ਵਧੇਰੇ ਬੁਨਿਆਦੀ ਮੁੱਦਾ ਇਨਕਲਾਬ ਦੇ ਸੰਕਲਪ ਦੀ ਪ੍ਰਸੰਗਕਤਾ ਦਾ ਨਹੀਂ ਸਗੋਂ ਰਾਸ਼ਟਰੀ ਖੁਦਮੁਖਤਿਆਰੀ ਦਾ ਹੈ। ਅੱਜ ਦੇ ਵਿਸ਼ਵ ਪ੍ਰਬੰਧ ਵਿੱਚ ਸਾਨੂੰ ਜ਼ਰੂਰ ਇਹ ਪੁੱਛਣਾ ਪਵੇਗਾ ਕਿ ਕੀ ਇਹ ਸੰਭਵ ਹੈ ਕਿ ਏਕੀਕ੍ਰਿਤ ਵਰਗਾਂ ਦਾ ਇੱਕ ਹਿੱਸਾ ਆਪਣੇ ਆਪ ਨੂੰ ਇਸ ਨਾਲੋਂ ਵੱਖਰਾ ਕਰਕੇ ਇੱਕ ਵੱਖਰੀ ਤਰਾਂ ਦੇ ਸਮਾਜਕ ਵਿਕਾਸ ਅਤੇ ਇਨਕਲਾਬੀ ਰੂਪ ਵਿੱਚ ਵੱਖਰੀ ਤਰਾਂ ਦੇ ਸਮੂਹਿਕ ਪ੍ਰੋਜੈਕਟ ਦੇ ਰਸਤੇ ਉੱਤੇ ਤੁਰ ਪਏ

ਜੇਕਰ ਸਾਮਵਾਦ ਨੂੰ ਸਿਰਫ਼ ਆਧੁਨਿਕੀਕਰਨ ਦੇ ਇੱਕ ਰਸਤੇ ਵਜੋਂ ਸਮਝਿਆ ਜਾਵੇ, ਜਿਵੇਂ ਪੱਛਮ ਆਮ ਤੌਰ ਤੇ ਸਮਝਦਾ ਹੈ ਤਾਂ ਸੋਵੀਅਤ ਯੂਨੀਅਨ ਦੇ ਢਹਿ-ਢੇਰੀ ਹੋਣ ਦਾ ਕਾਰਨ ਸਾਮਵਾਦ ਦੀ ਸਫ਼ਲਤਾ ਸੀ ਨਾ ਕਿ ਅਸਫ਼ਲਤਾ, ਕਿਉਂਕਿ ਇਹ ਤੇਜ਼ ਆਧੁਨਿਕੀਕਰਨ ਦਾ ਰਸਤਾ ਹੀ ਸੀ ਜਿਸ ਕਾਰਨ ਪੰਦਰਾਂ ਕੁ ਵਰ੍ਹੇ ਪਹਿਲਾਂ ਤੱਕ ਵੀ ਇਹ ਸਮਝਿਆ ਜਾਂਦਾ ਸੀ ਕਿ ਸੋਵੀਅਤ ਯੂਨੀਅਨ ਨੇ ਪੱਛਮ ਦੀ ਬਰਾਬਰੀ ਕਰ ਲਈ ਹੈ

Figure 6.7: Gurmukhi script text image

ਪੰਜਾਬੀ ਸਾਹਿਤ ਅਤੇ ਪੇਜਾਬ ਦੇ ਸਭਿਆਚਾਰ ਵਿੱਚ ਰੁਚੀ ਰਖੱਣ ਵਾਲੰ ਬੁੱਧਾਂਜੀਵੀਆਂ ਅਤੇ ਖੋਜੀਆਂ ਦੀਆਂ ਲੇਤਾਂ ਨੂੰ ਮੁੱਖ ਰਖਦੇ ਹੋਏ ਪੰਜਾਬੀ ਪ੍ਰੈਸ ਅਨੁਕ੍ਰਮਣਿਕਾਂ ਤਿਆਰ ਕੀਤੀ ਜਾਂਦੀ ਹੈ। ਇਬ ਸੁਯੀ ਵਿੱਰ ਪੰਜਾਬੀ ਰੇਟਰੈਂਸ ਲਾਇਬਰੇਰੀ ਵਿਬ ਪ੍ਰਾਪਤ ਹੋ ਰਹੇ ਰਸਾਲਿਆਂ ਅਤੇ ਅਖਬਾਰਾਂ ਵਿੱਬ ਛਪੇ ਚੇਣਵੇਂ ਲੇਖਾਂ ਦੀ ਇੰਡੈਕਸਿੰਗ ਕੀਤੀ ਗਈ ਹੇ।

ਅਨੁਕ੍ਰਮਣਿਕਾ ਦੀ ਤਰਤੀਬ ਲੇਖਾਂ ਦੋ ਇੰਦਰਾਜਾਂ ਨੂੰ ਵਿਸੇਛਾਰ ਤਰਤਾਂਬ ਦਿੱਤੀ ਗਈ ਹੈ। ਮੁੱਖ ਵਿਸ਼ੈ ਦੇ ਹੈਠਾਂ ਉਪ ਵਿਸ਼ੈ ਅਤੇ ਹਰ ਉਪ ਵਿਸ਼ੇ ਵਿੱਚ ਇੰਦਰਾਜਾਂ ਨੂੰ ਲੇਖਕਾਂ ਦੇ ਨਾਂ ਹੇਠ ਅਖੱਰ ਕ੍ਰਮ ਅਠਸਾਰ ਦਰਸਾਇਆ ਰਿਆ ਹੈ। ਸਾਰੇ ਇੰਦਰਾਜਾਂ ਨੂੰ ਕ੍ਰਮਵਾਰ ਇੰਦਰਾਜ ਨੰਬਰ ਦਿੱਤਾ ਗਿਆ ਹੈ ਜਿਵੇਂ ਆਦਿ। ਇਹ ਨੰਬਰ ਹਰ ਇੰਦਰਾਜ ਦੇ ਖੱਬੇ ਹੱਧ ਦਬਸਾਇਆ ਗਿਆ ਹੇ।

ਇੰਦਰਾਜਾਂ ਦੀ ਬਟਤਰ ਇੰਟਰਾਜਾਂ ਵਿੱਚ ਕ੍ਰਮਟੁਾਰ ਲੇਖਕ ਸਿਰਲੈਖ ਰਸਾਲੇ ਅਖਬਾਰ ਦਾ ਨਾਂ ਜਿਲਦ ਨੰਬਬ ਐਕ ਨੰਬਰ ਪ੍ਰਕਾਸਨ ਦੀ ਮਿਤੀ ਮਹੀਠਾ ਸਾਲ ਅਤੇ ਪਨੇ ਆਦਿ ਦਿੱਤੇ ਗਏ ਹਨ।

ਅਨੁਕ੍ਰਮਣਿਕ ਦੀ ਉਪਯੋਗਤਾ ਰਧਾਉਣ ਲਈ ਇਸ ਦੇ ਆਰੰਭ ਵਿੱਚ ਇੰਡੈਕਸ ਹੇ ਰਰੇ ਰਸਾਲਿਆਂ ਅਤੇ ਅਖਬਾਰਾਂ ਦੀ ਸੂਰੀ ਅਤੇ ਅੰਤ ਵਿਚ ਵਿਸ਼ਾ ਅਤੇ ਲੇਖਕ ਅਨੁਕ੍ਰਮਣਿਕ ਲਗਾਈ ਗਈ ਹੈ। ਵਿਸ਼ਾ ਅਨੁਕ੍ਰਮਣਿਕ ਟਿਚ ਦਰਸਾਏ ਹਰ ਵਿਸ਼ੇ ਦੇ ਸਾਹਮਟੇ ਅਤੇ ਲੇਖਕ ਅਠਕ੍ਰਮਣਿਕਾ ਵਿਚ ਹਰ ਲੇਖਗ ਦੇ ਨਾਂ ਸਾਹਮਣੇ ਸੰਥੰਧਤ ਇੰਦਰਾਜ ਨੰਧਰ ਦਰਸਾਏ ਰਾਦੇ ਹਨ। ਅਸੀਂ ਆਸ ਕਰਦੇ ਹਾਂ ਕਿ ਇਹ ਸੁਚੀ ਬੁੱਧੀਜਾਂਵੀਆਂ ਅਧਿਆਪਕਾਂ ਅਤੇ ਖੋਜਾਰੀਧੀਆਂ ਲਈ ਉਪਯੋਗੀ ਹੋਟੇਗੀ। ਪਾਠਕਾਂ ਦੇ ਸੁਝਾਵਾਂ ਲਈਂ ਅਸੀਂ ਰਿਣੀ ਹੇਵਾਂਗੇ।

Figure 6.8: OCR output of the image in Figure 1.4

ਪਤਮ ਕਰਨ ਵਾਲਾ ਟੀ ਜੋ ਦਰਅਸਲ ਮੰਗਾਂ ਦੀ ਇਕੱਤਰਤਾ ਹੈ, ਜਿਸਨੂੰ ਰੁਣਾਵੀ ਸੰਘਰਸ਼ ਵਿਚ ਖੇਬੰ ਪੱਖੀਆਂ ਦੀ ਜਿੱਤ ਜਾਂ ਥਸਤੀਵਾਦੀ ਸੱਤਾ ਦਾ ਸ਼ਾਤਮਾ ਬੁਰੂ ਕਰ ਸਕਦੇ ਹਨ ਪ੍ਰੰਤੁ ਇਸ ਤੇਂ ਪਿਛੇਂ ਜੋ ਵਿਸ਼ਾਲ ਤੇਂ ਇਸ਼ਾਲਤਮ ਜਨ ਅਧਾਰ ਅਤੋ ਇਨਕਲਾਥੀਕਰਨ ਦੇ ਸਰੁਪ ਡਿੱਚ ਵਟ ਸਕਦਾ ਹੈ। ਨਵੇਂ ਜਨ ਅਧਾਰਾਂ ਨਾਲ ਸੰਬੰਧਤ ਮੰਗਾਂ ਵਾਲੀਆਂ ਇਹ ਲਹਿਰਾਂ ਜੋ ਹੁਣ ਤੱਕ ਖਾਮੋਸ ਰਹੀ ਭੜੀ ਟੁੱਟੀ ਨੇਕਾਈ ਈਆਂ ਪਹਿਨਾਂ ਨਾਲੋ ਵੀ ਗਹਿਰੀਆਂ ਤਹਿਆਂ ਵਿੱਚੋਂ ਉਪਜਦੀਆਂ ਹਨ, ਮਹਿਚ ਦਿਖਾਵੋ ਲਈ ਖੇਬੰ ਪੱਖਾਂ ਹੋਣ ਦਾ ਦਾਅਵਾ ਕਰਨ ਾਲੀ ਸਰਕਾਰ ਦਾ ਵੀ ਇਨਕਲਾਥੀਕਰਨ ਕਰ ਸਕਦਾਂਆਂ ਹਨ ਅਤੇ ਰਾਜ ਵਿੱਚ ਪਹਿਲਾਂ ਨਾਲੋਂ ਵੱਧ ਫੇਸਲਾਕੁਨ ਤਧਦੀਲੀਆਂ ਸੰਰਵ ਕਰ ਵਿਖਾਉਂਦੀਆਂ ਹਨ। ਇਸ

Figure 6.9: OCR output of the image in Figure 1.5

6.5 Results and Discussion

The results of the experiments conducted by us are presented in this section. Table 6.2 shows the recognition accuracy by using SFAM neural network on structural features, while Table 6.3 gives the recognition accuracy by using SFAM neural network on statistical features**.** Table 6.4 shows the recognition accuracy by using SFAM neural network on combined structural and statistical features. Figure 6.10 contains the recognition accuracy of different features using SFAM classifier.

Table 6.2: Recognition accuracy of structural features using SFAM neural network classifier

Feature	Number of features	Recognition accuracy (%)
St1 to St10	18	16.08
St11	256	64.18
St11	128 (odd)	55.72
St11	128 (even)	62.19
St12	200 (M = 5, T = 5)	62.19
St12	140 (M = 5, T = 7)	54.73
St12	100 (M =5, T = 10)	51.74
St12	160 (M =4, T = 5)	62.69
St12	112 (M =4, T = 7)	55.72
St12	80 (M = 4, T = 10)	52.73
St12	120 (M = 3, T = 5)	61.19
St12	84 (M = 3, T = 7)	54.73
St12	60 (M = 3, T = 10)	48.78
Combined structural	416 [256 (St11)+160 (St12)]	66.67
Combined structural	376 [256 (St11) + 120 (St12)]	67.66
Combined structural	316 [256 (St11) + 60 (St12)]	69.65
Combined structural	248 [128 (St11) + 120 (St12)]	68.66
Combined structural	188 [128 (St11) + 60 (St12)]	66.67
Combined structural	334 [18 (St1 to St10) + 256 (St11) + 120 (St12)]	68.16
Combined structural	206 [18 (St1 to St10) + 128 (St11) + 60 (St12)]	68.66

Table 6.3: Recognition accuracy of statistical features using SFAM neural network classifier

Feature	Number of features	Recognition accuracy (%)
Zoning	144 (grid size = 4)	68.68
Zoning	100 (grid size = 5)	66.67
Zoning	64 (grid size = 6)	65.16
Zoning	49 (grid size = 7)	61.19
Zoning	36 (grid size = 8)	56.72
Zoning	25 (grid size = 10)	47.80
Zoning	16 (grid size = 12)	39.76
Moments (ZM)	54 (order = 13)	40.80
Moments (ZM)	88 (order = 17)	43.78
Moments (ZM)	130 (order = 21)	44.78
Moments (ZM)	180 (order = 25)	43.28
Moments (OFM)	53 (order = 9)	46.80
Moments (OFM)	76 (order = 11)	49.30
Moments (OFM)	103 (order = 13)	46.30
Combined statistical	306 [100 (zoning) + 130 (ZM) + 76 (OFMM)]	72.14
Combined statistical	270 [64 (zoning) + 130 (ZM) + 76 (OFMM)]	69.15
Combined statistical	255 [49 (zoning) + 130 (ZM) + 76 (OFMM)]	67.66
Combined statistical	228 [64 (zoning) + 88 (ZM) + 76 (OFMM)]	69.65
Combined statistical	205 [64 (zoning) + 88 (ZM) + 53 (OFMM)]	70.15
Combined statistical	186 [64 (zoning) + 88 (ZM) + 34 (OFMM)]	71.15
Combined statistical	152 [64 (zoning) + 54 (ZM) + 34(OFMM)]	70.65

Table 6.4: Recognition accuracy of all structural and statistical features using SFAM neural network classifier

Feature	Number of features	Recognition accuracy (%)
Combined (structural and statistical)	400 [64 (zoning) + 54 (ZM) + 34 (OFMM) + 128 (St11) + 120(St12)]	76.62
Combined (structural and statistical)	340 [64 (zoning) + 54 (ZM) + 34 (OFMM) + 128 (St11) + 60 (St12)]	74.64

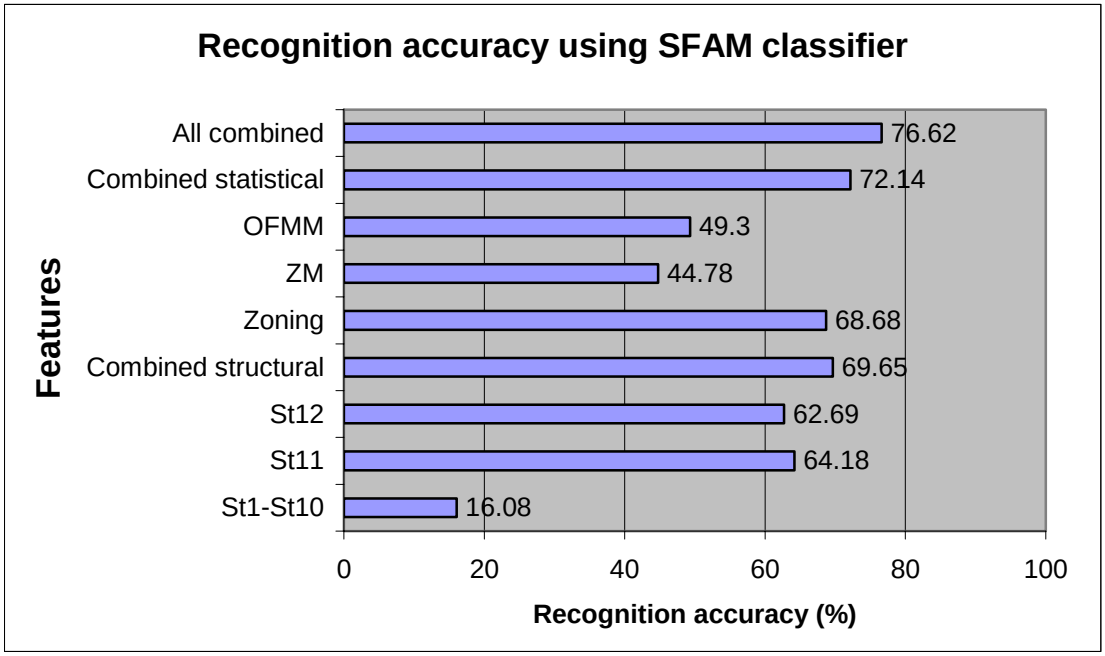


Figure 6.10: Recognition accuracy for different features using SFAM classifier

The results of *k*-NN classifier using various kinds of structural and statistical features and their various options have been tabulated in Tables 6.5-6.7. Table 6.5 shows the recognition accuracy using *k*-NN with different values of *k* on different sets of structural features. From Table 6.5, it can be observed that we get maximum accuracy of 83.60% at $k = 1$ when all the structural features have been used. It can also be observed from Table 6.5 that individually each structural feature does not perform well, but when used in combination, these start working effectively. Similarly, from Table 6.6, one can see that maximum accuracy of 82.59% has been achieved using statistical features at $k = 1$, when 306 feature vector size consisting of 100 features of zoning (grid size = 5), 130 features of Zernike moments (order 21) and 76 features of OFM moments (order 11) have been used.

The performance at k=3 is also comparable (or better) many times as compared to k=1. The reason for better performance at k=1 is that for higher vale of k, the voting method votes for wrong character as the class of confusing characters is placed in space near to the target character.

Table 6.7 shows the combined effect of statistical features and structural features on the accuracy of degraded printed Gurmukhi characters. One can note that a recognition accuracy of 86.09% has been achieved using a feature vector of size 400. Figures 6.11a-6.11c contains

the effect of different structural features on the recognition accuracy using *k*-NN classifier for different values of *k.*

Table 6.5: Recognition accuracy of structural features using *k*-NN classifier for different values of *k*

Feature	Number of features	Recognition accuracy (%)					
		***k* = 1**	***k* = 3**	***k* = 5**	***k* = 7**	***k* = 9**	***k* = 11**
St1 to St10	18	18.90	19.90	20.90	21.89	24.38	20.90
St11	256	69.65	70.15	68.66	65.67	66.67	63.68
St11	128 (odd)	58.71	60.20	58.71	61.19	59.20	57.71
St11	128 (even)	60.20	62.19	62.69	60.20	62.69	61.19
St12	200 (M = 5, T = 5)	73.63	74.63	70.15	68.66	66.67	65.57
St12	140 (M = 5, T = 7)	66.67	69.15	65.67	66.67	63.68	63.18
St12	100 (M =5, T = 10)	66.17	66.67	66.67	64.18	61.70	64.18
St12	160 (M =4, T = 5)	74.13	74.13	70.15	69.15	68.16	68.66
St12	112 (M =4, T = 7)	67.16	69.15	65.67	66.67	63.68	63.18
St12	80 (M = 4, T = 10)	66.67	66.67	66.67	64.18	61.20	64.18
St12	120 (M = 3, T = 5)	73.63	72.64	70.65	69.65	68.66	68.66
St12	84 (M = 3, T = 7)	68.66	69.15	64.68	66.67	63.68	63.18
St12	60 (M = 3, T = 10)	65.67	65.67	67.16	63.18	61.19	62.19
Combined structural	416 [256 (St11) + 160 (St12)]	80.60	80.10	77.11	75.12	77.11	76.62
Combined structural	376 [256 (St11) + 120 (St12)]	81.59	80.10	78.11	76.62	77.11	76.12
Combined structural	316 [256 (St11) + 60 (St12)]	78.61	78.11	79.60	76.12	72.14	71.64
Combined structural	248 [128 (St11) + 120 (St12)]	79.10	79.60	76.12	73.63	74.13	73.63
Combined structural	188 [128 (St11) + 60 (St12)]	79.10	79.10	77.11	73.63	74.63	75.12
Combined structural	206 [18 (St1 to St10) + 128 (St11-even) + 60 (St12)]	80.60	80.60	79.10	78.61	79.10	77.10
Combined structural	264 [18 (St1 to St10) + 128 (St11-even) + 120 (St12)]	82.59	81.60	81.09	79.11	79.11	77.60
Combined structural	394 [18 (St1 to St10) + 256 (St11) + 120 (St12)]	**83.10**	82.60	81.59	78.61	79.11	77.60

Table 6.6: Recognition accuracy of statistical features using *k*-NN classifier for different values of *k*

Feature	Number of features	Recognition accuracy (%)					
		***k* = 1**	***k* = 3**	***k* = 5**	***k* = 7**	***k* = 9**	***k* = 11**
Zoning	144 (grid size = 4)	72.64	71.14	70.15	71.64	69.65	66.67
Zoning	100 (grid size = 5)	74.13	73.63	71.14	70.65	68.16	65.17
Zoning	64 (grid size = 6)	72.14	67.16	64.18	65.17	62.19	62.19
Zoning	49 (grid size = 7)	68.16	68.16	67.66	63.68	61.19	59.20
Zoning	36 (grid size = 8)	63.08	65.57	67.16	63.18	62.19	62.19
Zoning	25 (grid size = 10)	58.21	58.71	60.70	60.70	60.70	58.71
Zoning	16 (grid size = 12)	43.78	43.78	43.78	43.28	39.30	43.28
Moments(ZM)	54 (order = 13)	44.23	43.78	44.73	43.78	42.73	43.78
Moments(ZM)	88 (order = 17)	48.71	47.21	46.72	45.77	46.72	45.27
Moments(ZM)	130 (order = 21)	49.70	47.21	46.22	45.27	46.72	46.22
Moments(ZM)	180 (order = 25)	48.20	47.70	47.70	46.70	47.20	45.70
Moments(OFM)	53 (order = 9)	62.19	62.69	56.67	56.67	57.66	56.67
Moments(OFM)	76 (order = 11)	67.17	65.62	61.19	63.68	66.17	64.68
Moments(OFM)	103 (order = 13)	62.19	63.68	62.19	65.17	62.69	64.68
Combined statistical	306 [100 (zoning) + 130 (ZM) + 76 (OFMM)]	**82.59**	81.60	80.09	81.59	80.09	79.11
Combined statistical	270 [64 (zoning) + 130 (ZM) + 76 (OFMM)]	79.60	80.60	80.09	78.61	77.11	78.61
Combined statistical	255 [49 (zoning) + 130 (ZM) + 76 (OFMM)]	76.62	76.62	76.11	76.11	75.12	74.13
Combined statistical	228 [64 (zoning) + 88 (ZM) + 76 (OFMM)]	79.60	77.11	73.63	72.64	73.63	72.14
Combined statistical	205 [64 (zoning) + 88 (ZM) + 53 (OFMM)]	81.09	76.62	74.13	74.64	74.13	73.13
Combined statistical	186 [64 (zoning) + 88 (ZM) + 34 (OFMM)]	81.09	76.12	75.12	73.13	73.13	72.14
Combined statistical	152 [64 (zoning) + 54 (ZM) + 34 (OFMM)]	81.09	76.12	75.12	73.13	73.13	72.14

Table 6.7: Recognition accuracy of all structural and statistical features using *k*-NN classifier for different values of *k*

Feature	Number of features	Recognition accuracy (%)					
		***k* = 1**	***k* = 3**	***k* = 5**	***k* = 7**	***k* = 9**	***k* = 11**
Combined (structural and statistical)	400 [64 (zoning) + 54 (ZM) + 34 (OFMM) + 128 (St11) + 120 (St12)]	**86.09**	85.10	81.59	77.60	76.62	75.62
Combined (structural and statistical)	340 [64 (zoning) + 54 (ZM) + 34 (OFMM) + 128 (St11) + 60 (St12)]	82.59	80.10	76.62	77.61	73.63	71.64

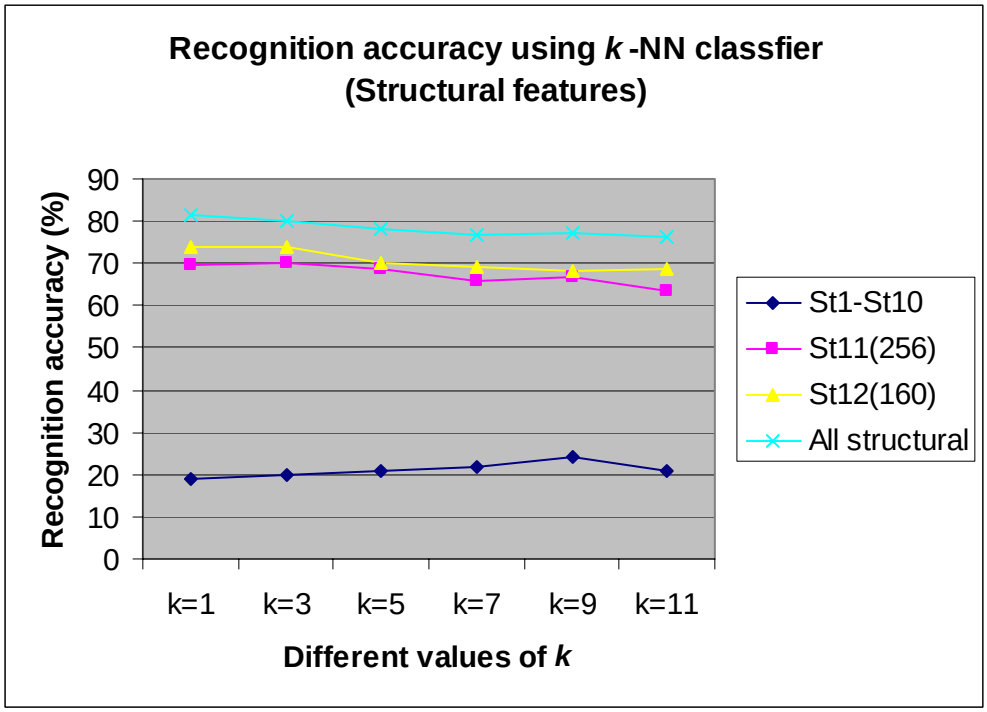


Figure 6.11a: Effect of different structural features on the recognition accuracy using *k*-NN classifier for different values of *k*

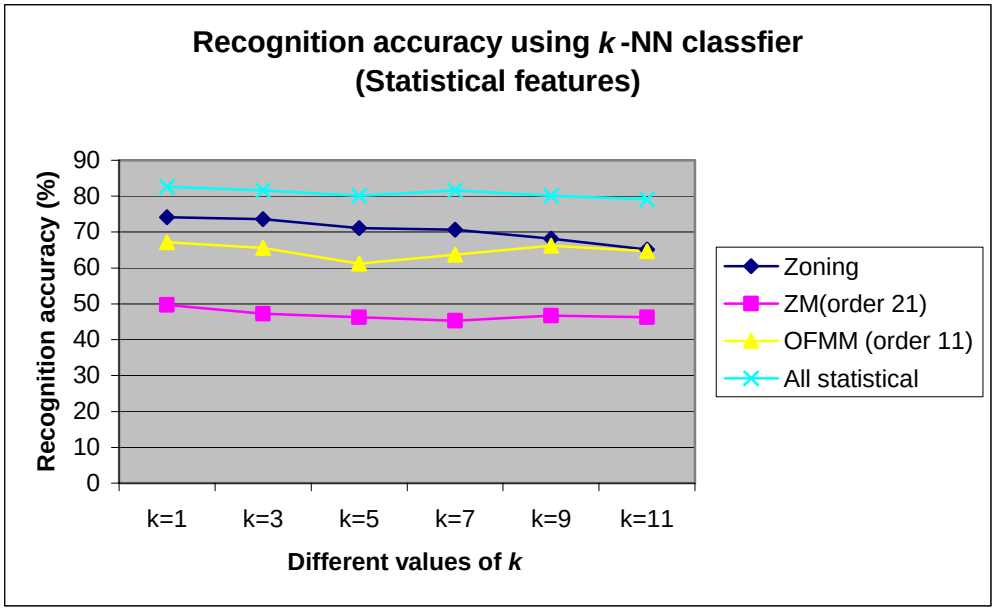


Figure 6.11b: Effect of different statistical features on the recognition accuracy using *k*-NN classifier for different values of *k*

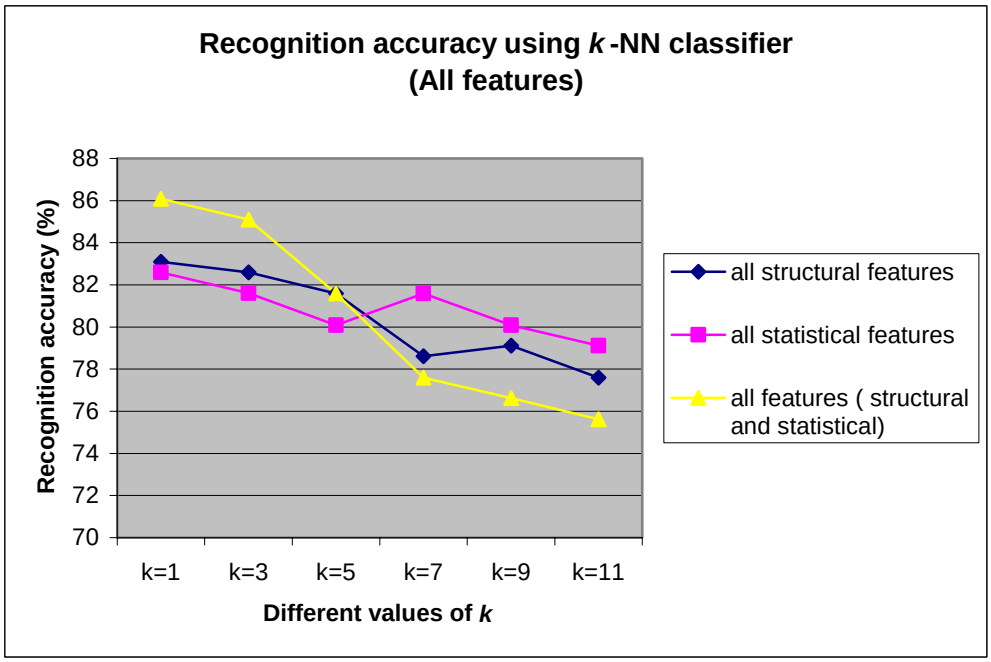


Figure 6.11c: Effect of all features on the recognition accuracy using *k*-NN classifier for different values of *k*

The results of SVM using various kinds of structural and statistical features and their various options are presented in Tables 6.8-6.10. Table 6.8 shows the recognition accuracy on different sets of structural features. From Table 6.8, it can be observed that we get maximum accuracy of 91.54% when all the structural features have been used. Similarly, from Table 6.9, one can see that maximum recognition accuracy of 88.58% has been achieved using statistical features only, when 228 feature vector size consisting of 64 features of zoning (grid size 8), 88 features of Zernike moments (order 13) and 76 features of OFM moments (order 11) have been used.

Table 6.10 shows the effect of combined statistical features and structural features on the recognition accuracy of degraded printed Gurmukhi characters as recognition accuracy of 92.54% has been observed using a feature vector of size 400. Figure 6.12 contains the recognition accuracy of different features using SVM classifier.

Table 6.8: Recognition accuracy of structural features using SVM classifier

Feature	Number of features	Recognition accuracy (%)
St1 to St10	18	24.05
St11	256	84.08
St11	128 (odd)	77.61
St11	128 (even)	79.10
St12	200 (M = 5, T = 5)	82.59
St12	140 (M = 5, T = 7)	73.13
St12	100 (M =5, T = 10)	64.68
St12	160 (M =4, T = 5)	83.08
St12	112 (M =4, T = 7)	76.12
St12	80 (M = 4, T = 10)	67.16
St12	120 (M = 3, T = 5)	81.59
St12	84 (M = 3, T = 7)	76.12
St12	60 (M = 3, T = 10)	61.19
Combined structural	416 [256 (St11) + 160 (St12)]	88.54
Combined structural	376 [256 (St11) + 120 (St12)]	89.55
Combined structural	316 [256 (St11) + 60 (St12)]	91.54
Combined structural	248 [128 (St11) + 120 (St12)]	90.54
Combined structural	188 [128 (St11) + 60 (St12)]	91.04
Combined structural	334 [18 (St1 to St10) + 256 (St11) + 120 (St12)]	90.05
Combined structural	206 [18 (St1 to St10) + 128 (St11) + 60 (St12)]	91.04

Table 6.9: Recognition accuracy of statistical features using SVM classifier

Feature	Number of features	Recognition accuracy (%)
Zoning	144 (grid size = 12)	87.06
Zoning	100 (grid size = 10)	84.08
Zoning	64 (grid size = 8)	82.09
Zoning	49 (grid size = 7)	69.65
Zoning	36 (grid size = 6)	66.17
Zoning	25 (grid size = 5)	65.17
Zoning	16 (grid size = 4)	50.49
Moments (ZM)	54 (order = 13)	47.12
Moments (ZM)	88 (order = 17)	53.73
Moments (ZM)	130 (order = 21)	57.21
Moments (ZM)	180 (order = 25)	55.72
Moments (OFM)	53 (order = 9)	59.70
Moments (OFM)	76 (order = 11)	63.18
Moments (OFM)	103 (order = 13)	59.20
Combined statistical	306 [100 (zoning) + 130 (ZM) + 76 (OFMM)]	84.08
Combined statistical	270 [64 (zoning) + 130 (ZM) + 76 (OFMM)]	88.58
Combined statistical	255 [49 (zoning) + 130 (ZM) + 76 (OFMM)]	80.10
Combined statistical	228 [64 (zoning) + 88 (ZM) + 76 (OFMM)]	88.58
Combined statistical	205 [64 (zoning) + 88 (ZM) + 53 (OFMM)]	85.57
Combined statistical	186 [64 (zoning) + 88 (ZM) + 34 (OFMM)]	86.07
Combined statistical	152 [64 (zoning) + 54 (ZM) + 34 (OFMM)]	85.57

Table 6.10: Recognition accuracy of all structural and statistical features using SVM classifier

Feature	Number of features	Recognition accuracy (%)
Combined (structural and statistical)	400 [64 (zoning) + 54 (ZM) + 34 (OFMM) + 128 (St11) + 120 (St12)]	92.54
Combined (structural and statistical)	340 [64 (zoning) + 54 (ZM) + 34(OFMM)+ 128 (St11) + 60 (St12)]	92.04

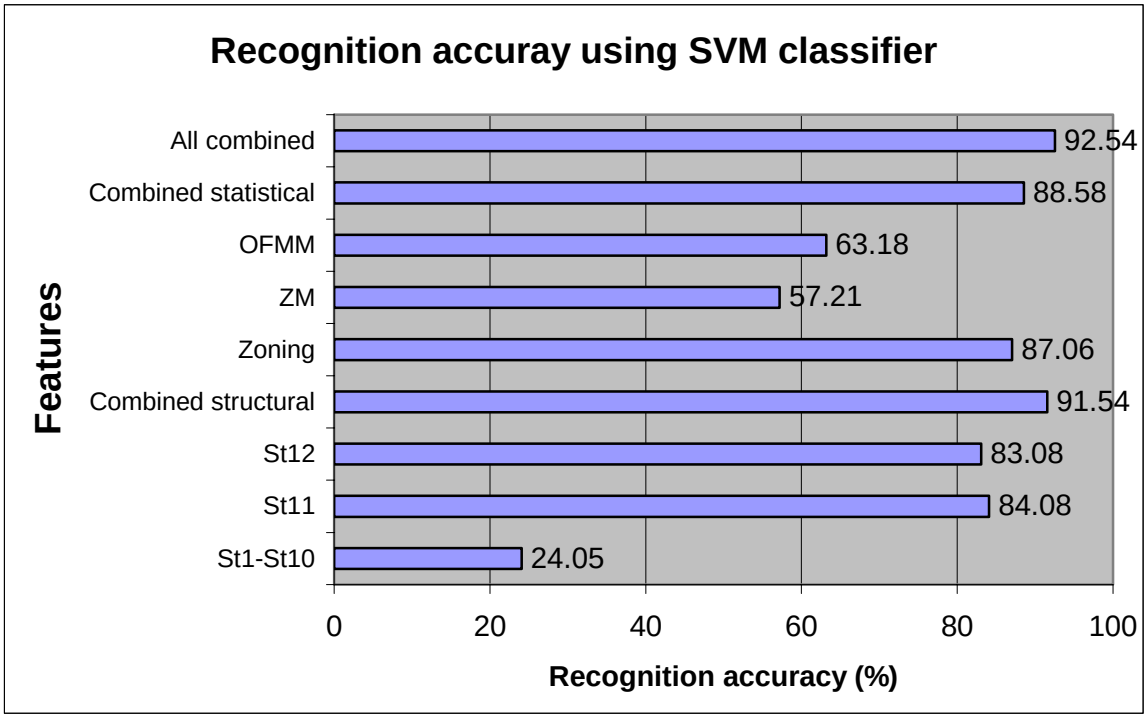


Figure 6.12: Recognition accuracy of different features using SVM classifier

From Table 6.11 and Figure 6.13, it can be inferred that a recognition accuracy of 92.54% for Gurmukhi characters has been achieved with SVM, which should be considered as best classifier for recognising degraded printed Gurmukhi characters and is without applying any post-processing operations. The Nearest neighbor classifier (*k*-NN) with *k* as 1 produces a recognition accuracy of 86.09% for recognising degraded printed Gurmukhi characters. A recognition accuracy of 76.62% has been achieved with neural networks for the recognition of degraded printed Gurmukhi characters.

Table 6.11: Recognition accuracy of different features using different classifiers

Classifiers	**Features**		
	Structural	**Statistical**	**Combined**
ANN(SFAM)	69.65	72.14	76.62
k-NN	83.10	82.59	86.09
SVM	91.54	88.58	**92.54**

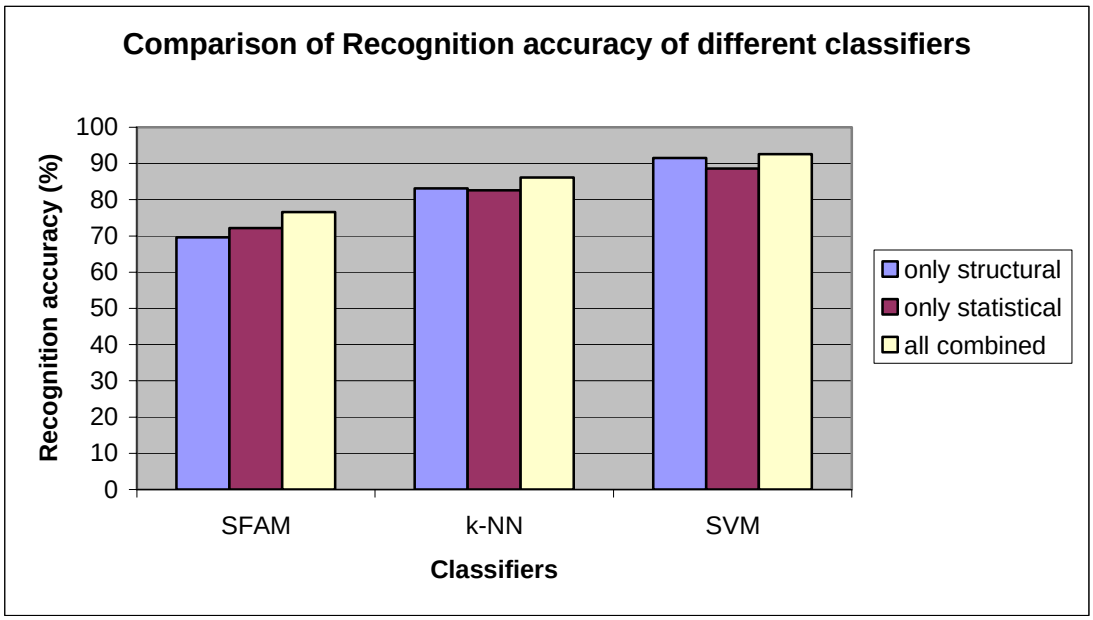


Figure 6.13: Comparison of recognition accuracy of different features using different classifiers

Similarly, we have given the recognition accuracy obtained using different features in upper zone and lower zone. The recognition accuracy using SFAM, *k*-NN and SVM on different set of features for upper zone has been given in Table 6.12 and that for lower zone in Table 6.13. One can see from these tables that recognition accuracy of 93.76% has been achieved in upper zone using SVM classifier while the recognition accuracy for lower zone is limited to 88.54% using SVM classifier.

Table 6.12: Recognition accuracy using different classifiers on different sets of features for upper zone

Feature	Number of features	Percentage accuracy		
		SFAM	***k*-NN**	**SVM**
St6 to St10	13	19.18	23.98	26.37
St11	256	75.06	80.81	82.49
St11	128 (odd)	71.46	79.86	81.29
St11	128 (even)	70.50	79.13	81.53
St12	200 (M = 5, T = 5)	63.79	75.54	80.58
St12	160 (M = 4, T = 5)	59.23	73.38	74.58
St12	120 (M = 3, T = 5)	57.55	73.62	75.06
Combined structural	416 [256 (St11) + 160 (St12)]	86.57	91.61	92.57
Combined structural	376 [256 (St11) + 120 (St12)]	85.61	89.44	90.65
Zoning	144 (grid size = 4)	61.15	72.42	75.30
Zoning	100 (grid size = 5)	63.31	74.34	78.65
Zoning	64 (grid size = 6)	60.43	73.38	76.74
Moments(ZM)	88 (order = 17)	30.46	45.56	46.76
Moments(ZM)	130 (order = 21)	34.77	46.52	49.88
Moments(ZM)	180 (order = 25)	27.58	32.13	41.25
Moments(OFM)	53 (order = 9)	40.29	56.35	60.78
Moments(OFM)	76 (order = 11)	46.80	60.43	63.79
Moments(OFM)	103 (order = 13)	32.85	41.00	49.88
Combined statistical	306 [100 (zoning) + 130 (ZM) + 76 (OFMM)]	64.51	78.66	82.73
Combined statistical	270 [64 (zoning) + 130 (ZM) + 76 (OFMM)]	67.39	81.29	85.85
Combined statistical	255 [49 (zoning) + 130 (ZM) + 76 (OFMM)]	72.42	77.22	81.29
Combined statistical	228 [64 (zoning) + 88 (ZM) + 76 (OFMM)]	75.30	82.49	85.85
Combined (structural and statistical)	400 [64 (zoning) + 54 (ZM) + 34 (OFMM) + 128 (St11) + 120 (St12)]	88.49	92.09	93.76

Table 6.13: Recognition accuracy using different classifiers on different sets of features for lower zone

Feature	Number of features	Recognition accuracy (%)		
		SFAM	***k*-NN**	**SVM**
St6 to St10	13	13.21	16.29	23.79
St11	256	61.67	71.37	82.82
St11	128 (odd)	65.64	76.21	84.14
St11	128 (even)	56.82	70.48	77.09
St12	200 (M = 5, T = 5)	52.42	61.23	64.76
St12	160 (M = 4, T = 5)	51.10	62.43	69.60
St12	120 (M = 3, T = 5)	50.22	61.23	67.40
Combined structural	416 [256 (St11) + 160 (St12)]	60.35	78.41	84.58
Combined structural	376 [256 (St11) + 120 (St12)]	58.15	77.97	81.94
Zoning	144 (grid size = 4)	43.61	57.27	59.47
Zoning	100 (grid size = 5)	45.81	58.59	63.44
Zoning	64 (grid size = 6)	43.17	56.39	61.67
Moments(ZM)	88 (order = 17)	23.34	34.80	40.09
Moments(ZM)	130 (order = 21)	29.95	36.45	41.85
Moments(ZM)	180 (order = 25)	26.43	32.15	36.12
Moments(OFM)	53 (order = 9)	34.80	42.29	45.37
Moments(OFM)	76 (order = 11)	37.88	44.49	48.46
Moments(OFM)	103 (order = 13)	31.71	41.41	43.17
Combined statistical	306 [100 (zoning) + 130 (ZM) + 76 (OFMM)]	49.78	65.64	76.65
Combined statistical	270 [64 (zoning) + 130 (ZM) + 76 (OFMM)]	50.22	68.72	78.41
Combined statistical	255 [49 (zoning) + 130 (ZM) + 76 (OFMM)]	49.34	63.44	74.45
Combined statistical	228 [64 (zoning) + 88 (ZM) + 76 (OFMM)]	48.46	62.11	70.48
Combined (structural and statistical)	400 [64 (zoning) + 54 (ZM) + 34 (OFMM) + 128 (St11) + 120 (St12)]	67.84	71.37	88.54

Table 6.14 contains the confusion matrix for all Gurmukhi sub-symbols. One can see from the table that many sub-symbols shows 100% recognition accuracy. The additional consonants in middle zone have less recognition accuracy due to degradation of *bindī.* Also, upper zone and lower zone sub-symbols shows less recognition accuracy.

Table 6.14: Confusion matrix for Gurmukhi characters

Characters	Recognised characters (Percentage of recognition accuracy)					Recognition accuracy (%)
ਉ	ਉ (93.33)	ਚ (6.67)				93.33
ਓ	ਓ (72.72)	ਉ (27.28)				72.72
ਅ	ਅ (100)					100
ਏ	ਏ (91.30)	ਝ (8.70)				66.67
ਸ	ਸ (86.67)	ਸ਼ (6.67)	ਬ (3.33)	ਯ (3.33)		86.67
ਹ	ਹ (87.5)	ਰ (7.5)	ਚ (5)			87.5
ਕ	ਕ (90)	ਦ (7.5)	ਏ (2.5)			90
ਖ	ਖ (81.81)	ਥ (13.64)	ਪ (4.55)			81.81
ਘ	ਘ (100)					100
ਙ	ਙ (100)					100
ਚ	ਚ (77.78)	ਭ (11.11)	ਰ (11.11)			77.78
ਛ	ਛ (86.66)	ਡ(6.67)	ਭ (6.67)			86.66
ਜ	ਜ (100)					100
ਝ	ਝ (90)	ਚ (10)				90
ਞ	ਞ (100)					100
ਟ	ਟ (90)	ਏ (10)				90
ਠ	ਠ (93.33)	ਨ (6.67)				93.33
ਡ	ਡ (100)					100
ਢ	ਢ (70)	ਚ (20)	ਏ (10)			70
ਣ	ਣ (92)	ਟ (8)				92
ਤ	ਤ (89.19)	ੜ (8.11)	ਭ (2.70)			89.19
ਥ	ਥ (90.90)	ਚ (4.55)	ਭ (4.55)			54.54
ਦ	ਦ (73.33)	ਏ (10)	ਚ (10)	ਜ਼ (6.67)		73.33
ਧ	ਧ (100)					100
ਨ	ਨ (86.68)	ਲ (6.67)	ਠ (6.67)			86.67
ਪ	ਪ (100)					100
ਫ	ਫ (87.50)	ਫ਼ (6.25)	ਛ (6.25)			87.50
ਬ	ਬ (92.50)	ਥ (5)	ਧ (2.5)			92.50
ਭ	ਭ (93.10)	ਡ (6.90)				93.10
ਮ	ਮ (96.55)	ਪ (3.45)				96.55
ਯ	ਯ (100)					100
ਰ	ਰ (100)					100
ਲ	ਲ (87.50)	ਨ (12.50)				100
ਵ	ਵ (77.5)	ਵ (12.5)	ਏ (5.0)	ਡ (2.5)	ਟ (2.5)	77.5
ੜ	ੜ (85)	ਡ (10)	ਤ (5)			85
ਸ਼	ਸ਼ (60)	ਸ (20)	ਬ (20)			60
ਜ਼	ਜ਼ (75)	ਜ (10)	ਭ (10)	ਚ (5)		75

ਖ	ਖ (83.34)	ਖ(8.33)	ਸ (8.33)			83.33
ਫ਼	ਫ਼ (57.14)	ਫ (42.86)				57.14
ਗ਼	ਗ਼ (50)	ਘ (50)				50
ਲ਼	ਲ਼ (85)	ਲ (15)				85
ਾ	ਾ (96)	missing (4)				90
ਿ	ਿ(98.04)	ੀ (0.98)	ਿ (0.98)			98.04
ੀ	ੀ (97.06)	ਿ(1.96)	ਿ (0.98)			97.06
ੁ	ੁ(89.61)	ੁ(5.19)	ੂ(2.60)	ੂ (2.60)		89.61
ੂ	ੂ(88.68)	ੁ(5.66)	ੁ(3.77)	ੂ(1.89)		88.68
ੇ	ੁ(90.82)	ੁ(4.08)	ੂ(3.06)	ੂ(2.04)		90.82
ੈ	ੂ(94.56)	ੁ(5.46)				94.56
ੋ	ੋ (87)	missing (13)				87
ੌ	ੌ (88.63)	ੁ(5.68)	ੋ (3.40)	੍ (2.27)		88.63
੍	੍ (79.59)	ੁ(10.21)	ੂ(7.14)	ੂ(3.06)		79.59
ਂ	ਂ (92.5)	missing (5)	ਂ (2.5)			92.5
ਃ	ਃ(77.14)	ਂ (17.15)	ਂ (5.71)			77.14
ਂ	ਂ (81.81)	੍ (9.09)	ਃ(5.45)	missing(3.64)		81.81
੍	੍ (100)					100
ੜ	ੜ (90)	ਂ(10)				90

Chapter 7
Conclusions

A complete solution for recognising texts for degraded printed Gurmukhi script documents has been proposed in this thesis. Many kinds of degradations are found in Gurmukhi documents, especially in old books. Degradation may be due to defects in paper, defects introduced during printing, defects introduced during digitization and/or defects introduce during copying. Different kinds of degradations available in Gurmukhi script documents have been discussed by us in this thesis. Sources of each kind of degradation, problems associated with each kind of degraded text, comparison of each kind of degraded text in Gurmukhi script with the corresponding degraded text in Roman script, some possible solutions for identifying each kind of degraded text have also been discussed. Two kinds of degradations, namely, touching characters and heavily printed characters have been discussed in detail and their solutions have been proposed in this work. The reason of working on these two kinds of degradation is that touching characters and heavily printed characters are found in all kinds of degradations like faxed documents, typewritten documents and backside text visible documents.

Mainly, three stages including segmentation, feature extraction and classification have been explored in detail for recognising degraded printed Gurmukhi script documents in this work. New algorithms have been proposed for segmenting overlapping lines and for segmenting touching characters. First algorithm proposed in this thesis segments the overlapping lines along with associating the broken components of a line to that line for Gurmukhi documents. The algorithm has been developed on the basis of a heuristic. Second algorithm has been developed for segmenting overlapping lines and associating the broken components of a line to that line for Indian scripts, including, Devanagari, Bangla, Gujarati, Kannada, Tamil, Telugu and Malayalam scripts. Third algorithm has been proposed for

segmenting overlapping lines involving different sized text. These algorithms can also be used for developing an OCR for Indian script newspapers.

Based on the structural properties of Gurmukhi script characters, the fourth algorithm has been designed for segmenting touching sub-symbols in upper zone. The idea of aspect ratio used in literature for identifying touching characters fails most of the times for upper zone characters. On the basis of the concept of concavity and convexity found for upper zone sub-symbols, this algorithm is capable of segmenting even vowel *bindī* from other sub-symbols in upper zone. The fifth and last algorithm has been proposed for segmenting touching characters in middle zone. Again, this algorithm is based on the structural properties of the Gurmukhi script sub-symbols in middle zone. Instead of looking for candidate of segmentation, touching points are searched on the basis of structural properties like full and half sidebars in the sub-symbols. Furthermore, this algorithm also solves the problem of segmentation of more than two touching sub-symbols in a single word and the problem of finding exact segmentation column, if the width of touching blob is greater than or equal to the width of the stroke.

Structural and statistical features have been used for extracting features of the Gurmukhi sub-symbols. Along with ten important and traditionally used structural features, Directional Distance Distribution and transitions features have been used first time for recognising an Indian script characters.

The neural networks, *k*-nearest neighbor and Support Vector Machines have been used for classification purposes.

7.1 Contributions of the Work

It is believed that this thesis has made the following contributions in the field of Indian script recognition:

- This is the first reported work to recognise a degraded printed Indian script.
- In Chapter 3, various kinds of degradation available in printed Gurmukhi script have been discussed. This is first of its kind of reported work for an Indian script. These

various kinds of degradations are also available in other Indian scripts. As such, it opens a wide area of research for people engaged in recognition of Indian scripts.

- New algorithms have been developed for segmenting horizontally overlapping lines and associating the broken components of a line to their respective lines in degraded printed Gurmukhi script, in Chapter 4. Different types of strips have been recognised and complete solution has been provided for segmentation of horizontally overlapping line and associating small strips to their respective line. The algorithm proposed for Gurmukhi script is also working efficiently for other major Indian scripts for the same purpose. This is first of its kind of reported work dealing with all major Indian scripts. This work is very important for developing OCR for Indian script newspapers.
- The algorithms designed for segmentation of touching characters in upper and middle zone use the structural properties of the Gurmukhi script. The idea can be extended to segment touching characters in other Indian scripts, especially other headline based scripts like Devanagari and Bangla.
- The structural and statistical features selected for recognition of degraded printed Gurmukhi script are robust to noise and touching blobs of the characters. Directional Distance Distribution feature and transitions feature have been used first time for recognising Indian script documents.

7.2 Future Work

We feel that this work has opened up many areas of research that can be explored in future. Some of the areas are identified in the following paragraphs.

The current work can be expanded in future for recognition of Gurmukhi documents containing other kinds of degradation like broken characters, faxed documents, typewritten documents and backside text visible documents. The work can also be extended to recognition of other Indian scripts containing these kinds of degradations.

Use of Gabor filters and Hidden Markov Models can also be explored in for recognising degraded printed documents for Gurmukhi script, in particular, and for other Indian scripts, in general. Post-processing methods can be implemented to enhance the recognition accuracy of the system proposed in this thesis.

Bibliography

[1] S. N. Srihari and S. W. Lam, "Character recognition", Center of Excellence for Document Analysis and Recognition (CEDAR), *Technical Report*, 1995.

[2] M. E. Stevens, "Automatic character recognition-State-of-the-art report", National Bureau of Standards & Technology, Tech. Note 112, Washington, USA, 1961.

[3] S. Mori, C. Y. Suen and K. Yamamoto, "Historical review of OCR research and development", *Proceedings of the IEEE,* Vol. 80(7), pp. 1029-1058, 1992.

[4] U. Pal and B. B. Chaudhuri, "Indian script character recognition: a survey", *Pattern Recognition*, Vol. 37(9), pp. 1887-1899, 2004.

[5] S. Mori, K. Yamamoto and M. Yasuda, "Research on machine recognition of hand-printed characters", *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI),* Vol. 6(4), pp. 386–405, 1984.

[6] G. Nagy, "At the frontiers of OCR", *Proceedings of the IEEE*, Vol. 80(7), pp. 1093-1100, 1992.

[7] R. Plamondon and S. N. Srihari, "On-line and off-line handwritten recognition: a comprehensive survey", *IEEE Transactions on PAMI*, Vol. 22(1), pp. 63–84, 2000.

[8] C. Y. Suen, R. Legault, C. Nadal, M. Cheriet and L. Lam, "Building a new generation of handwriting recognition systems", *Pattern Recognition Letters*, Vol. 14(4), pp. 305-315, 1993.

[9] U. Pal and B. B. Chaudhuri, "Computer recognition of printed Bangla script", *International Journal of Systems Science*, Vol. 26, pp. 2107-2123, 1995.

[10] U. Pal and B. B. Chaudhuri, "Printed Devanagari script OCR system", *Vivek*, Vol. 10(1), pp. 12-24, 1997.

[11] G. S. Lehal and C. Singh, "A complete machine printed Gurmukhi OCR system", *Vivek*, Vol. 16(3), pp. 10-17, 2006.

[12] M. Bosker, "Omnidocument technologies", *Proceedings of the IEEE*, Vol. 80(7), pp. 1066-1078, July 1992.

[13] T. Pavlidis, "Problems in recognising poorly printed text", in Symposium on Document Analysis and Information Retrieval(SDAIR), pp. 163-172, 1992.

[14] S. V. Rice, F. R. Jenkins and T. A. Nartker, "The fourth annual test of OCR accuracy", Technical Report 95-04, ISRI, University of Nevada, Las Vegas, pp. 11-50, 1995.

[15] J. Rocha and T. Pavlidis, "A solution to the problem of touching and broken characters", in the Proceedings of the 2nd International Conference on Document Analysis and Recognition (ICDAR), pp. 602-605, 1993.

[16] C. Fang, *Deciphering Algorithms for Degraded Document Recognition*, Ph. D. thesis, State University of New York at Buffalo, 1997.

[17] H. Bunke and P. S. P. Wang, *Handbook of Character Recognition and Document Image Analysis*, World Scientific Publishing Company, 1997.

[18] Stephen V. Rice, George Nagy and Thomas A. Nartker, *Optical Character Recognition: An Illustrated Guide to the Frontier*, Kluwer Academic Publications, 1999.

[19] S. Mori, H. Nishida and H. Yamada, *Optical Character Recognition*, John Wiley & Sons, 1999.

[20] J. Mantas, "An overview of character recognition methodologies", *Pattern Recognition*, Vol. 19(6), pp. 425-430, 1986.

[21] V. K. Govindan and A. P. Shivaprasad, "Character recognition-A review", *Pattern Recognition*, Vol. 23 (7), pp. 671-683, 1990.

[22] C. Y. Suen, M. Berthod and S. Mori, "Automatic recognition of hand printed characters- the state of the art", *Proceedings of the IEEE*, Vol. 68(4), pp. 469-487, 1980.

[23] S. Impedovo, L. Ottaviano and S. Occhinegro, "Optical character recognition- a survey", *International Journal Pattern Recognition and Artificial Intelligence*, Vol. 5(1-2), pp. 1-24, 1991.

[24] Q. Tian. P. Zhang. T. Alexander and Y. Kim, "Survey: omnifont-printed character recognition", in the proceedings of Visual Communications and Image Processing SPIE, Vol. 1606, pp. 260-268, 1991.

[25] A. K. Jain, R. P. W. Duin and J. Mao, "Statistical pattern recognition: a review", *IEEE Transactions on PAMI*, Vol. 22(1), pp. 4-37, 2000.

[26] R. Kasturi and L. O'Gorman, "Document image analysis: a bibliography", *Machine Vision and Applications*, Vol. 5(3), pp. 231-243, 1992.

[27] C. C Tappert, C. Y. Suen and T. Wakahara, "The state of the art in on-line handwriting recognition", *IEEE Transactions on PAMI*, Vol. 12(8), pp. 787-808, 1990.

[28] T. Wakahara, H. Murase and K. Odaka, "On-line handwriting recognition", *Proceedings of the IEEE*, Vol. 80(7), pp. 1181-1194, 1992.

[29] F. Nouboud and R. Plamondon, "On-line recognition handprinted characters: survey and beta tests", *Pattern Recognition*, Vol. 23(9), pp. 1031-1044, 1990.

[30] S. D. Connell, R. M. K. Sinha and A. K. Jain, "Recognition of unconstrained **o**n-line Devanagari characters", in the Proceedings of 15th International Conference on Pattern Recognition (ICPR), Vol. 2, Spain, pp. 368-371, 2000.

[31] S. D. Connell and A. K. Jain, "Template-based online character recognition", *Pattern Recognition*, Vol. 34(1), pp. 1-14, 2001.

[32] F. Bortolozzi, A. Britto Jr., L. S.Oliveria and M. Morita, "Recent advances in handwriting recognition", in the Proceedings of International Workshop on Document Analysis (IWDA), India, pp. 1-30, 2005.

[33] S. W. Lee, "Off-line recognition of totally unconstrained handwritten numerals Using multiplayer cluster neural network", *IEEE Transactions on PAMI*, Vol. 18(6), pp. 648-652, 1996.

[34] F. El-Khaly and M. A. Sid-Ahmed, "Machine recognition of optically captured machine printed Arabic text", *Pattern Recognition*, Vol. 23(11), pp. 1207-1214, 1990.

[35] A. Amin, "Off-line Arabic character recognition- a survey", in the Proceedings of 4th ICDAR, pp. 596-599, 1997.

[36] L. M. Lorigo and V. Govindaraju, "Offline Arabic handwriting recognition: a survey", *IEEE Transactions on PAMI*, Vol. 28(5), pp. 712-724, 2006.

[37] T. H. Hildebrandt and W. Liu, "Optical recognition of handwritten Chinese characters: Advances since 1980", *Pattern Recognition*, Vol. 26(2), pp. 205-225, 1993.

[38] C. L. Liu, S. Jaeger and Masaki Nakagawa, "Online recognition of Chinese characters: the state-of-the-art", *IEEE Transactions on PAMI*, Vol. 26(2), pp. 198-213, 2004.

[39] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation", *IEEE Transactions on PAMI*, Vol. 18(7), pp. 690-706, 1996.

[40] C. E. Dunn and P. S. P. Wang, "Character segmentation techniques for handwritten text - a survey", in the Proceedings of 11th ICPR, Vol. 2, pp. 577-580, 1992.

[41] Y. Lu, "Machine printed character segmentation - an overview", *Pattern Recognition*, Vol. 28(1), pp. 67-80, 1995.

[42] Y. Lu and M. Shridhar, "Character segmentation in handwritten words - an overview", *Pattern Recognition*, Vol. 29(1), pp. 77-96, 1996.

[43] R. L. Hoffman and J. W. McCullough, "Segmentation methods for recognition of machine-printed characters", *IBM Journal of Research and Development*, Vol. 15(2), pp. 153-165, 1971.

[44] H. S. Baird, S. Kahan and T. Pavlidis, "Components of an omnifont page reader", in the Proceedings of 8th ICPR, Paris, pp. 344-348, 1986.

[45] S. Tsujimoto and H. Asada, 1992, "Major components of a complete text reading system", *Proceedings of the IEEE*, Vol. 80(7), pp. 1133-1149, 1992.

[46] S. Liang, M. Shridhar and M. Ahmadi, "Segmentation of touching characters in printed document recognition", *Pattern Recognition*, Vol. 27(6), pp. 825-840, 1994.

[47] J. Wang and J. S. N. Jean, "Segmentation of merged characters by neural networks and shortest path", *Pattern Recognition*, Vol. 27(5), pp. 649-658, 1994.

[48] R. G. Casey and G. Nagy, "Recursive segmentation and classification of composite character patterns", in the Proceedings of 6th ICPR, pp. 1023-1026, 1982.

[49] S. W. Lee, D. J. Lee and H. S. Park, "A new methodology for gray-scale character segmentation and recognition", *IEEE Transactions on PAMI*, Vol. 18(10), pp. 1045-1050, 1996.

[50] S. Tsujimoto and H. Asada, "Resolving ambiguity in segmenting touching characters", in the Proceedings of 1st ICDAR, pp. 701-709, 1991.

[51] T. Hong, *Degraded Text Recognition using Visual and Linguistic Context*, Ph. D. thesis, Computer Science Department of SUNY at Buffalo, 1995.

[52] S. Kahan, T. Pavlidis and H. S. Baird, "On the recognition of printed characters of any font and size", *IEEE Transactions on PAMI*, Vol. 9(2), pp. 274-288, 1987.

[53] C. B. Bose and S. S. Kuo, "Connected and degraded text recognition using hidden markov model", *Pattern Recognition*, Vol. 27(10), pp. 1345-1363, 1994.

[54] M. Schenkel and M. Jabri, "Low resolution, degraded document recognition using neural networks and hidden markov models", *Pattern Recognition Letters*, Vol. 19(3-4), pp. 365-371, 1998.

[55] S. Zhao, Z. Chi, P. Shi and H. Yan, "Two-stage segmentation of unconstrained handwritten Chinese characters", *Pattern Recognition*, Vol. 36(1), pp. 145-156, 2003.

[56] O. Nakamura, M.Ujiie, N.Okamoto and T. Minami, "A character segmentation algorithm for mixed-mode communication", *Trans. IEICE*, (D) 167-D, 11, pp. 1277-1285, 1984.

[57] A. Elnagar and R. Alhajj, "Segmentation of connected handwritten numeral strings", *Pattern Recognition*, Vol. 36(3), pp. 625-634, 2003.

[58] Y. Donggang and Y. Hong, "Separation of single-touching handwritten numeral strings based on structural features", *Pattern Recognition*, Vol. 31(12), pp. 1835-1847, 1998.

[59] Y. Donggang and Y. Hong, "Separation of touching handwritten multi-numeral strings based on morphological structural features", *Pattern Recognition*, Vol. 34(3), pp. 587-599, 2001.

[60] Z. Chi, M. Suters and H. Yan, "Separation of single- and double-touching handwritten numeral strings", *Optical Engineering*, Vol. 34(4), pp. 1159-1165, 1995.

[61] Z. Lu, Z. Chi, W. C. Siu and P. Shi, "A background-thinning-based approach for separating and recognising connected handwritten digit strings", *Pattern Recognition*, Vol. 32(6), pp. 921-933, 1999.

[62] U. Pal, A. Belaïd and Ch. Choisy, "Touching numeral segmentation using water reservoir concept", *Pattern Recognition Letters*, Vol. 24(1-3), pp. 261-272, 2003.

[63] Y. -K. Chen and J. -F. Wang, "Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis", *IEEE Transactions on PAMI*, Vol. 22(11), pp. 1304-1317, 2000.

[64] V. Bansal and R. M. K. Sinha, "Segmentation of touching and fused Devanagari characters", *Pattern Recognition*, Vol. 35(4), pp. 875-893, 2002.

[65] U. Garain and B. B. Chaudhuri, "Segmentation of touching characters in printed Devanagari and Bangla scripts using fuzzy multifactorial analysis", *IEEE Transactions on Systems, Man and Cybernetics*, Part C, Vol. 32(4), pp. 449–459, 2002.

[66] U. Garain and B. B. Chaudhuri, "On recognition of touching characters in printed Bangla documents", in the Proceedings of Indian Conference on Computer Vision, Graphics, and Image Processing, pp. 377-380, 1998.

[67] G. S. Lehal and C. Singh, "Text segmentation of machine-printed Gurmukhi script", Document Recognition and Retrieval VIII, proceedings SPIE, USA, Vol. 4307, pp. 223-231, 2001.

[68] G. S. Lehal and C. Singh, "A technique for segmentation of Gurmukhi text", Computer Analysis of Images and Patterns, Proceedings CAIP 2001, W. Skarbek (Ed.), Lecture Notes in Computer Science, Vol. 2124, Springer-Verlag, Germany, pp. 191-200, 2001.

[69] B. B. Chaudhuri, U. Pal and M. Mitra, "Automatic recognition of printed Oriya script", in the Proceedings of 6th ICDAR, pp. 795-799, 2001.

[70] V. Bansal, *Integrating Knowledge Sources in Devanagari Text Recognition*, Ph. D. thesis, IIT Kanpur, India, 1999.

[71] S. Harikumar, K. Jithesh, K. G. Sulochana and R. Ravindra Kumar, "Script based line and character segmentation techniques for Malayalam document images", in the Proceedings of International Symposium on MT, NLP and Translation Support Systems (iSTRANS 2004), New Delhi, India, pp. 122-127, 2004.

[72] U. Pal and S. Datta, "Segmentation of Bangla unconstrained handwritten text", in the Proceedings of 7th ICDAR, pp. 1128-1132, 2003.

[73] U. Pal, S. Sinha and B. B. Chaudhuri, "Multi-script line identification from Indian documents", in the Proceedings of 7th ICDAR, pp. 880-884, 2003.

[74] U. Pal and B. B. Chaudhuri, "Automatic separation of machine-printed and hand-written text lines", in the Proceedings of 5th ICDAR, pp. 645-648, 1999.

[75] J. Dholakia, A. Negi and S. R. Mohan, "Zone identification in the printed Gujarati text", in the Proceedings of 8th ICDAR, pp. 272-276, 2005.

[76] B. B. Chaudhuri and U. Pal, "A complete printed Bangla OCR system", *Pattern Recognition,* Vol. 31(5), pp. 531-549, 1998.

[77] G. S. Lehal, *Optical Character Recognition of Machine Printed Gurmukhi Text***,** Ph. D. thesis, Punjabi University, Patiala, India, 2001.

[78] C. Y. Suen, "Character recognition by computer and applications", in Handbook of Pattern Recognition and Image Processing, New York: Academic, pp. 569-586, 1986.

[79] O. D. Trier, A. K. Jain and T. Taxt, "Feature extraction methods for character recognition: - a survey", *Pattern Recognition,* Vol. 29(4), pp. 641-662, 1996.

[80] L. Heutte, T. Paquet, J. V. Moreau, Y. Lecourtier and C. Olivier, "A structural/statistical feature based vector for handwritten character recognition", *Pattern Recognition Letters,* Vol. 19(7), pp. 629-641, 1998.

[81] T. H. Reiss, "The revised fundamental theorem of moment invariants", *IEEE Transaction on PAMI,* Vol. 13(8), pp. 830-834, 1991.

[82] S. O. Belkasim, M. Shridhar and M. Ahmadi, "Pattern recognition with moment invariants: a comparative study and new results", *Pattern Recognition,* Vol. 24(12), pp. 1117-1138, 1991.

[83] J. Flusser and T. Suk, "Affine moment invariants: a new tool for character recognition", *Pattern Recognition Letters*, Vol. 15(4), pp. 433-436, 1994.

[84] M. K. Hu, "Visual pattern recognition by moment invariants", *IRE Transactions on Information Theory,* Vol. 8(2), pp. 179-187, 1962.

[85] S. S. Reddi, "Radial and angular moment invariants for image identification", *IEEE Transactions on PAMI*, Vol. 3(2), pp. 240-242, 1981.

[86] C. H. Teh and R. T. Chin, "On image analysis by the method of moments", *IEEE Transactions on PAMI,* Vol. 10(4), pp. 496-513, 1988.

[87] A. B. S. Hussain, G. T. Toussaint and R. W. Donaldson, "Results obtained using a simple character recognition procedure on Munson's handprinted data", *IEEE Transactions on Computers,* pp. 201-205, 1972.

[88] S. Messelodi and C. M. Modena, "Context driven text segmentation and recognition", *Pattern Recognition Letters,* Vol. 17(1), pp. 47-56, 1996.

[89] R. Tarling and R. Rohwer, "Efficient use of training data in the *n*-tuple recognition method", *Electronics Letters*, Vol. 29(24), pp. 2093-2094, 1993.

[90] M. H. Glauberman, "Character recognition for business machines", *Electronics*, Vol. 29, pp. 132-136, 1956.

[91] T. Akiyama and N. Hagita, "Automated entry systems for printed documents", *Pattern Recognition*, Vol. 23(11), pp. 1141-1154, 1990.

[92] H. J. Lee and B. Chen, "Recognition of handwritten Chinese characters via short line segments", *Pattern Recognition,* Vol. 25(5), pp. 543-552, 1992.

[93] A. Amin, "Recognition of printed Arabic text based on global features and decision tree learning techniques", *Pattern Recognition,* Vol. 33, pp. 1309-1323, 2000.

[94] L. L. Lee and N. R. Gomes, "Disconnected handwritten numeral image recognition", in the Proceedings of 4th ICDAR, pp. 467-470, 1997.

[95] J. Rocha and T. Pavlidis, "A shape analysis model with applications to a character recognition system", *IEEE Transactions on PAMI,* Vol. 16(4), pp. 393-404, 1994.

[96] G. Leedham and V. Pervouchine, "Validating the use of handwriting as a biometric and its forensic analysis", in the Proceedings of International Workshop on Document Analysis (IWDA), India, pp. 175-192, 2005.

[97] M. Shridhar and A. Badreldin, "High accuracy character recognition using Fourier and topological descriptors", *Pattern Recognition,* Vol. 17(5), pp. 515-524, 1984.

[98] C. L. Liu and H. Fujisawa, "Classification and learning for character recognition: comparison of methods and remaining problems", in the Proceedings of Neural Networks and Learning in Document Analysis and Recognition (NNLDAR-2005), pp. 1-7, 2005.

[99] B. A. Draper, C. E. Brodley and P. E. Utgoff, "Goal-directed classification using linear machine decision trees", *IEEE Transactions on PAMI,* Vol. 16(9), pp. 888-893, 1994.

[100] A. Kundu, Y. He and M. Y. Chen, "Alternatives to variable duration HMM in handwriting recognition", *IEEE Transactions on PAMI,* Vol. 20(11), pp. 1275-1280, 1998.

[101] A. K. Jain, J. Mao and K. M. Mohiuddin, "Artificial neural networks: a tutorial", *Computer,* pp. 31-44, 1996.

[102] Y. L. Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition", *Neural Compution*, Vol. 1, pp. 541-551, 1989.

[103] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition", *Knowledge Discovery and Data Mining*, Vol. 2(2), pp. 121-167, 1998.

[104] L. S. Oliveira and R. Sabourin, "Support vector machines for handwritten numerical string recognition", in the Proceedings of 9th International Workshop on Frontiers in Handwriting Recognition(IWFHR), pp. 39-44, 2004.

[105] Y. S. Huang and C.Y. Suen, "A method of combining multiple experts for the recognition of unconstrained handwritten numerals", *IEEE Transactions on PAMI*, Vol. 17(1), pp. 90-94, 1995.

[106] H. S. Baird, "Feature identification for hybrid structural/statistical pattern classification", *Computer Vision, Graphics, and Image Processing*, Vol. 42(3), pp. 318-333, 1988.

[107] R. M. K. Sinha and H. N. Mahabala, "Machine recognition of Devanagari script", *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 9, pp. 435-441, 1979.

[108] R. M. K. Sinha, "Rule based contextual post-processing for Devanagari text recognition", *Pattern Recognition*, Vol. 20(5), pp. 475-485, 1987.

[109] R. M. K. Sinha, "Role of context in Devanagari script recognition", *J. Inst. Elec. Telecom. Engg.*, Vol. 33(3), pp. 86-91, 1987.

[110] I. K. Sethi and B. Chatterjee, "Machine recognition of constrained hand-printed Devanagari numerals", *J. Inst. Elec. Telecom. Engg*., Vol. 22, pp. 532-535, 1976.

[111] I. K. Sethi and B. Chatterjee, "Machine recognition of constrained hand printed Devanagari", *Pattern Recognition*, Vol. 9(2), pp. 69-76, 1977.

[112] S. Palit and B. B. Chaudhuri, "A feature-based scheme for the machine recognition of printed Devanagari script", in the Proceedings of Indian Conference on Pattern Recognition, Image Processing and Computer Vision, pp. 163-168, 1995.

[113] K. K. Biswas and S. Chatterjee, "Feature based recognition of Hindi characters", in the Proceedings of Indian Conference on Pattern Recognition, Image Processing and Computer Vision, pp. 182-187, 1995.

[114] A. K. Ray and B. Chatterjee, "Design of a nearest neighbor classifier system for Bengali character recognition", *J. Inst. Elec. Telecom. Engg.*, Vol. 30, pp. 226-229, 1984.

[115] A. K. Dutta, "A generalized formal approach for description and analysis for major Indian scripts", *J. Inst. Elec. Telecom. Engg.*, Vol. 30, pp. 155-161, 1984.

[116] A. Dutta and S. Chaudhury, "Bengali alpha-numeric character recognition using curvature features", *Patter Recognition*, Vol. 26(12), pp. 1757-1770, 1993.

[117] B. B. Chaudhuri and U. Pal, "Skew angle detection of digitized Indian script documents", *IEEE Transactions on PAMI*, Vol. 19(2), pp. 182-186, 1997.

[118] U. Garain and B. B. Chaudhuri, "Compound character recognition by run number based metric distance", SPIE Proc., Vol. 3305, pp. 90-97, 1998.

[119] U. Pal and S. Datta, "Segmentation of Bangla unconstrained handwritten text", in the Proceedings of 7th ICDAR, pp. 1128-1132, 2003.

[120] U. Bhattacharya, T. K. Das, A. Datta, S. K. Parui and B. B. Chaudhuri, "A hybrid scheme for handprinted numeral recognition based on a self-organizing network and MLP classifiers", *International Journal on Pattern Recognition and Artificial Intelligence*, Vol. 16(7), pp. 845-864, 2002.

[121] S. Sural and P. K. Das, "An MLP using Hough transform based fuzzy feature extraction for Bengali script recognition", *Pattern Recognition Letters*, Vol. 20, pp. 771-782, 1999.

[122] G. Siromoney, R. Chandrasekaran and M. Chandrasekaran, "Machine recognition of printed Tamil characters", *Pattern Recognition*, Vol. 10, pp. 243-247, 1978.

[123] M. Chandrasekaran, R. Chandrasekaran and G. Siromoney, "Context dependent recognition of handprinted Tamil characters", in the Proceedings of International Conference on Systems Man and Cybernetics, Vol. 2, pp. 786-790, 1983.

[124] P. Chinnuswamy and S. G. Krishnamoorthy, "Recognition of hand-printed Tamil characters", *Pattern Recognition*, Vol. 12(3), pp. 141-152, 1980.

[125] K. H. Aparna, V. Subramaniam, M. Kasirajan, G. V. Prakash, V. S. Chakravarthy and S. Madhvanath, "Online handwrting recognition for Tamil", in the Proceedings of 9th International Workshop on Frontiers in Handwriting Recognition(IWFHR), pp. 438-443, 2004.

[126] S. N. S Rajasekaran and B. L. Deekshatulu, "Recognition of printed Telugu characters", *Computer Graphics and Image Processing (CGIP)*, Vol. 6, pp. 335-360, 1977.

[127] C. V. Lakshmi and C. Patvardhan, "A high accuracy OCR system for printed Telugu text", in the Proceedings of Conference on Convergent Technologies for Asia-Pacific Region (TENCON 2003), Vol. 2, pp. 725-729, 2003.

[128] A. Negi, C. Bhagvati and B. Krishna, "An OCR system for Telugu", in the Proceedings of the Sixth International Conference on Document Processing, pp. 1110-1114, 2001.

[129] S. Mohanti, "Pattern recognition in alphabets of Oriya language using kohonen neural network", *Int. J. Pattern Recogn. Artif. Intell.*, Vol. 12, pp. 1007-1015, 1998.

[130] K. Roy, T. Pal, U. Pal and F. Kimura, "Oriya handwritten numeral recognition system", in the Proceedings of ICDAR, pp. 770-774, 2005.

[131] G. S. Lehal and C. Singh, "Feature extraction and classification for OCR of Gurmukhi script", *Vivek*, Vol. 12(2), pp. 2-12, 1999.

[132] G. S. Lehal and C. Singh, "A post processor for Gurmukhi OCR", *Sadhana Academy Proceedings in Engineering Sciences*, Vol. 27(1), pp. 99-112, 2002.

[133] G. S. Lehal, C. Singh and R. Lehal, "A shape based post processor for Gurmukhi OCR", in the Proceedings of 6^{th} ICDAR, pp. 1105-1109, 2001.

[134] G. S. Lehal and P. Singh, "A technique for segmentation of machine printed Gurmukhi script", in the Proceedings of 4th International Conference on Cognitive Systems, Delhi, India, pp. 283-287, 1998.

[135] A. Goyal, G. S. Lehal and S. S. Deol, "Segmentation of machine printed Gurmukhi script", in the Proceedings of 9^{th} International Graphonomics Society Conference, Singapore, pp. 293-297, 1999.

[136] M. K. Jindal, G. S. Lehal and R. K. Sharma, "Segmentation problems and solutions in printed degraded Gurmukhi script", *International Journal of Signal Processing*, Vol. 2(4), pp. 258-267, 2005.

[137] M. K. Jindal, G. S. Lehal and R. K. Sharma, "A study of touching characters in degraded Gurmukhi text", *Transactions on Engineering, Computing and Technology* , ENFORMATIKA, Vol. 4, pp. 121-124, 2005.

[138] M. K. Jindal, G. S. Lehal and R. K. Sharma, "A study of touching characters in degraded Gurmukhi script", in the proceedings of International Conference on Pattern Recognition and Computer Vision, PRCV 2005, WEC (2) 2005, Istanbul, Turkey, pp. 121-124, 2005.

[139] M. K. Jindal, R. K. Sharma and G. S. Lehal, "Segmentation of horizontally overlapping lines in printed Gurmukhi script", in the Proceedings of 14th International Conference on Advanced Computing and Communications (ADCOM 2006), pp. 226-229, published by IEEE Computer Society Press, NJ USA, 2006.

[140] M. K. Jindal, G. S. Lehal and R. K. Sharma, "Segmentation of touching characters of Indian scripts-an overview", in the proceedings of National Conference on Recent Advances and Future Trends in IT (RAFIT 2005), Punjabi University Patiala, pp. 74-77, 2005.

[141] M. K. Jindal, R. K. Sharma and G. S. Lehal, "A study of different kinds of degradation in printed Gurmukhi script", in Proceedings of International Conference on Computing: Theory and Applications (ICCTA'07), pp. 538-544, Published by IEEE Computer Society, LA, USA, 2007.

[142] S. Antani and L. Agnihotri, "Gujarati character recognition", in the Proceedings of 5th ICDAR, pp. 418–421, 1999.

[143] A. Yajnik and S. R. Mohan, "Identification of Gujarati characters using wavelets and neural networks", in the Proceedings of International conference on Artificial Intelligence and Soft Computing, pp. 150-155, 2006.

[144] T. V. Ashwin, P. S. Sastry, "A font and size independent OCR system for printed Kannada documents using support vector machines", *Sadhana*, Vol. 27, pp. 35-58, 2002.

[145] R. S. Rao and R. D. S. Samuel, "On-line character recognition for handwritten Kannada characters using wavelet features and neural classifier", *IETE Journal of Research*, Vol. 46, pp. 387-392, 2000.

[146] B. V. Kumar and A. G. Ramakrishnan, "Machine recognition of printed Kannada text", in the Proceedings of 5th International Workshop on Document Analysis and Systems (DAS), pp. 37-48, 2002.

[147] N. Sharma, U. Pal and F. Kimura, "Recognition of handwritten Kannada numerals", in the Proceedings of 9th International Conference on Information Technology (ICIT'06), pp. 133-136, 2006.

[148] V. Vaidya, V. Deshpande, D. Gada and B. Shirole, "Statistical approach to feature extraction for numeral recognition from degraded documents", in the Proceedings of 5th ICDAR, Banglore, pp. 112-115, 1999.

[149] Y. Li, D. Lopresti, G. Nagy and A. Tomkins, "Validation of image defect models for optical character recognition", *IEEE Transactions on PAMI*, Vol. 18(2), pp. 99-108, 1996.

[150] H. S. Baird, "The state of the art of document image degradation modeling", invited talk, in the Proceedings of International Workshop on Document Analysis Systems, Rio de Janeiro, Brazil, pp. 10-13, 2000.

[151] T. Kanugou, *Document Degradation Model and Methodology for Degradation Model Validation*, Ph. D. thesis, Dept. EE, Univ. Washington, March 1996.

[152] http://en.wikipedia.org/wiki/Official_languages_of_India

[153] Y. Lu, B. Haist, L. Harmon, J. Trenkle and R. Vogi, "An accurate and efficient system for segmenting machine-printed text", U.S. Postal Service 5th Advanced Technology Conference, Washington D.C., Vol. 3, pp. A 93-A 105, 1992.

[154] N. Okamoto, O. Nakamura and T. Minami, "Character segmentation for mixed-mode communication", IFIP'83, pp. 681-685, 1983.

[155] B. A. Yanikoglu, "Pitch-based segmentation and recognition of dot-matrix text", *International Journal of Document Analysis and Recognition (IJDAR)*, Vol. 3, pp. 34-39, 2000.

[156] M. Droettboom, "Correcting broken characters in the recognition of historical printed documents", in the Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital libraries (JCDL), Houston, Texas, USA, pp. 364-366, 2003.

[157] Z. M. Kovacs, "A System for reading low quality characters from printouts", in the Proceedings of the 13th International Conference on Pattern Recognition, pp. 185-189, 1996.

[158] M. Oguro, T. Akiyama and K. Ogura, "Faxed document image restoration using gray level representation", in the Proceedings of 4th ICDAR, Vol. 2, pp. 679-683, 1997.

[159] T. R. Randolph and M. J. T. Smith, "Enhancement of fax documents using a binary angular representation", in the Proceedings of International Symposium on Intelligent Multimedia, Video and Speech Processing, Hong Kong, pp. 125-128, 2001.

[160] J. D. Hobby and T. K. Ho, "Enhancing degraded documents images via bitmap clustering and averaging", in the Proceedings of 4th ICDAR, pp. 394-400, 1997.

[161] P. Natarajan, I. Bazzi, Z. Lu, J. Makhoul and R. Scwhartz, "Robust OCR of degraded documents", in the Proceedings of 5th ICDAR, pp. 357-361, 1999.

[162] M. Cannon, J. Hochberg and P. Kelly, "QUARC: a remarkably effective method for Increasing the OCR accuracy of degraded typewritten documents", in the Proceedings of the 1999 Symposium on Document Image Understanding Technology (SDIUT'99), Annapolis, MD, pp. 154-158, May 1999.

[163] M. Cannon, J. Hochberg and P. Kelly, "Quality assessment and restoration of typewritten document images", *IJDAR*, Vol. 2(2-3), pp. 80-89, 1999.

[164] C. Rodríguez, J. Muguerza, M. Navarro, A. Zárate, J.I. Martín and J.M. Pérez, "Segmentation of low-quality typewritten digits", in the Proceedings of 14th ICPR, pp. 1106-1109, 1998.

[165] S. lliescu, R. Shinghal and R. Yee-Mian Teo, "Proposed heuristic procedures to preprocess character patterns using line adjacency graphs", *Pattern Recognition*, Vol. 29(6), pp. 951-975, 1996.

[166] T. J. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns", *Communications of the ACM*, Vol. 27(3), pp. 236-239, 1984.

[167] Il-Seok Oh and C. Y. Suen, "Distance features for neural network-based recognition of handwritten characters", *IJDAR*, Vol. 1(2), pp. 73-88, 1998.

[168] S. W. Lee and J. S. Park, "Nonlinear shape normalization methods for the recognition of large set handwritten characters", *Pattern Recognition*, Vol. 27(5), pp. 895-902, 1994.

[169] P. D. Gader, M. Mohamed and J. H. Chiang, "Handwritten word recognition with character and inter-character neural networks", *IEEE Transactions on Systems, Man, and Cybernetics-Part B:Cybernetics*, Vol. 27(1), pp. 158-164, 1997.

[170] A. Khotanzad and Y. H. Hong, "Invariant image recognition by Zernike moments", *IEEE Transactions on PAMI*, Vol. 12(5), pp. 489-497, 1990.

[171] C. Singh, “Improved quality of reconstructed images using floating point arithmetic for moment calculation”, *Pattern Recognition*, Vol. 39(11), pp. 2047-2064, 2006.C.

[172] Y. Sheng and L. Shen, “Orthogonal Fourier-Mellin moments for invariant pattern recognition”, *J. Optical Society of America,* Vol. 11(6), 1748-1757, 1994.

[173] C. Kan and M. D. Srinath, “Invariant Character Recognition with Zernike and Orthogonal Fourier-Mellin Moments”, *Pattern Recognition,* Vol. 35**,** pp. 143-154, 2002.

[174] T. Kasuba, “Simplified fuzzy ARTMAP”, *IEEE AI Expert*, pp. 19-25, 1993.

[175] N. Chaikla and Y. Qi, “ Genetic Algorithms in Feature Selection”, in the Proceedings of International Conference on Systems, Man and Cybernetics, pp. 538-540, 1999.

Publications Based on the Work Presented in this Thesis

Papers in refereed journals (published/communicated):

1. **M. K. Jindal**, G. S. Lehal and R. K. Sharma, "Segmentation Problems and Solutions in Printed Degraded Gurmukhi Script", *International Journal of Signal Processing*, Vol. 2(4), pp. 258-267, 2005

2. **M. K. Jindal**, G. S. Lehal and R. K. Sharma, "A Study of Touching Characters in Degraded Gurmukhi Text", *Proceedings of World Academy of Science, Engineering and Technology*, Vol. 4, pp. 121-124, 2005.

3. **M. K. Jindal**, G. S. Lehal and R. K. Sharma, "On Segmentation of touching characters and overlapping lines in degraded printed Gurmukhi script", *International Journal of Image and Graphics (IJIG)*, World Scientific Publishing Company.(**Revised version communicated on 30.10.2006**)

4. **M. K. Jindal**, R. K. Sharma and G. S. Lehal, "Segmentation of Horizontally Overlapping Lines in Printed Indian Scripts", *International Journal of Computational Intelligence and Research (IJCIR)*. **(In Press)**

Papers in conference proceedings (full length):

5. **M. K. Jindal**, R. K. Sharma and G. S. Lehal, "Structural Features for Recognizing Degraded Printed Gurmukhi Script", accepted for publication in Proceedings of the IEEE International Conference on Information Technology and New Generations (ITNG 2008), Las Vegas, USA, April 2008

6. **M. K. Jindal**, R. K. Sharma, G. S. Lehal, "A Study of Different Kinds of Degradation in Printed Gurmukhi Script", in the Proceedings of International Conference on Computing: Theory and Applications (ICCTA'07), pp. 538-544, Published by IEEE Computer Society Press, LA USA, March 2007.

7. **M. K. Jindal**, R. K. Sharma, G. S. Lehal, "Segmentation of Horizontally Overlapping lines in Printed Gurmukhi Script", in the Proceedings of 14th International Conference on Advanced Computing and Communications (ADCOM 2006), pp. 226-229, Published by IEEE Computer Society Press, NJ USA, December 2006.

8. **M. K. Jindal**, G. S. Lehal and R. K. Sharma, "Segmentation of touching characters of Indian Scripts-an Overview", in the Proceedings of National Conference on Recent Advances and Future Trends in IT (RAFIT 2005), Punjabi University Patiala, pp. 74-77, March 2005.

9. **M. K. Jindal**, G. S. Lehal, R. K. Sharma and Simpel Jindal, "Segmentation of touching characters in printed Gurmukhi Script", in the Proceedings of National Conference on Research & Practices in Current Areas of IT (RPIT 2004), SHSL-CIET Longowal, pp. 158-162, March 2004.